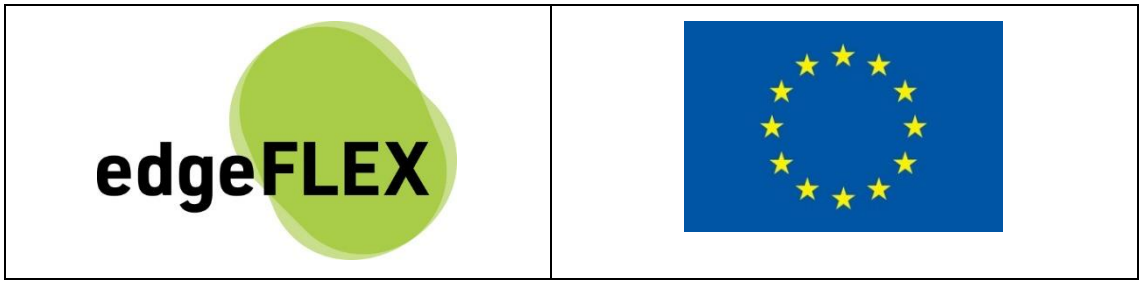

edgeFLEX

D3.3 Report on VPP Optimisation, V2

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no 883710.

Project Name	edgeFLEX
Contractual Delivery Date:	31.03.2022
Actual Delivery Date:	31.03.2022
Author(s):	ALPIQ
Workpackage:	WP3 – Optimisation of a VPP consisting of variable and dispatchable RES
Security:	P
Nature:	R
Version:	V1.0
Total number of pages:	**52**

Abstract
In this report, we explore the optimisation of virtual power plants (VPP), whose goal is to balance a wind park consisting of a portfolio of biogas power plants and a battery, while maximising the VPP revenues. Machine learning methods are employed to handle uncertainty in price and wind production. For price modeling, we consider the latest trends in the field and the most up-to-date events affecting the day-ahead and intra-day prices. The performance of our price models is demonstrated by both statistical methods and improvements in the profits of the virtual power plant.

Optimisation methods will take price and imbalance forecasts as input and conduct parallelisation, decomposition, and splitting methods to handle a sufficiently high number of assets of a VPP to operate the methods. The focus is on increasing the speed of computing optimal solutions to large-scale mixed-integer linear programming problems. The highest increase in computing speed enabled by our method. which we called Gradual Increase. is two orders of magnitude. Further progress in this methodology and its new facets are demonstrated.

Keyword list
Virtual Power Plant, Gradual Increase, Modified Gradual Increase, Proximal Jacobian ADMM, Cold Warmstart, Warm Warmstart

Disclaimer
All information provided reflects the status of the EDGEFLEX project at the time of writing and may be subject to change.

Executive Summary

In this report, we consider balancing a wind energy fleet with a portfolio of flexible assets: biogas-fired power plants and batteries. The original purpose of the biogas-fired power plants that we are considering in the VPP was to earn on the power exchange, primarily on the day-ahead market. When these assets become part of the VPP, their task changes as follows: maximising profits from the sale of electricity while ensuring the balancing of the wind energy fleet, which also results in the participation in the intra-day market. This task is challenging in terms of optimisation and price forecasting methods.

Two aspects are considered: optimisation and price forecasts. The former is a mixed integer optimisation and sophisticated decomposition methods are developed; in the latter we conduct several forecasting methods based on machine learning with the goal of maximising the resulting cumulative revenue. We also utilise commercial forecasts and our forecasting methods turned out to be competitive with these forecasts in terms of the resulting cumulative revenue. We also pay special attention to the robustness of the VPP to handle large amounts of assets within it. Besides, robust model predictive control (RPMC) is utilised to take many scenarios into account during decision making.

Authors

Partner	Name	e-mail
ALPIQ		
	Vadim Omelčenko	vadim.omelcenko@alpiq.com
	Ronan Kavanagh	ronan.kavanagh@alpiq.com

Table of Contents

1. Introduction

The optimisation of virtual power plants (VPP) is of crucial importance as it enables the more efficient incorporation of distributed energy resources (DER) into the grid and thereby contributes to the achievement of goals associated with ecology. Solar and wind energy sources have a degree of uncertainty and require constant balancing. Therefore, from an ecological point of view, it is desirable that the balancing is carried out by flexible energy sources operating on a renewable fuel. In our case studies, we consider the balancing of wind parks by means of pools of biogas (renewable fuel) power plants and pools of batteries and propose ways to accelerate calculations by means of mathematical methods of decomposition and splitting, leveraging the special structure of problems, parallelisation, and the brute force power of optimisation *solvers*. In our VPP, state-of-the-art commercial wind forecasts are used to determine the amounts of energy that our wind turbines can produce, and the goal of the flexible assets is to handle the aggregate imbalances in both directions: biogas power plants can balance only deficits/shortages while batteries can also balance surpluses.

The goal of the biogas power plants and batteries is to maximise their revenues from selling electricity on the day-ahead and intra-day markets while providing balancing to the pool of intermittent assets. There is special emphasis on the profitability of the assets because otherwise, the participation in the virtual power plant will not be attractive. Price forecasts and optimisation are two indispensable tools for ensuring that this goal will be achieved. Special attention is devoted to the ability of the VPP to scale i.e., its robustness to handle increasing numbers of assets.

Before taking up the main topic, it is necessary to mention progress in this field. One important research component within the optimisation of VPPs is the method known as Alternating Direction Method of Multipliers (ADMM), which is a technique based on replacing the solution of a large-scale optimisation problem with an iterative procedure involving solutions of many small subproblems.

A special facet of this method known as Proximal Jacobian ADMM is proven to o(1/k)-converge for linear programming problems and to be amenable to parallelisation [8]. This algorithm will also be used in our research. In this study, this method for optimizing large pools of power plants where we deal with integer variables is applied.

Before taking up the main topic of this deliverable, we mention relevant research in the field of optimising Virtual Power Plants. The scope of optimisation methods for managing virtual power plants is relatively broad and incorporates approaches such as linear programming, mixed-integer linear programming, methods based on dynamic programming, multistage bilevel algorithms, evolutionary algorithms, etc. [28]. In [29], a virtual power plant is optimised by linear programming methods and the polytope of the constraints is approximated with a zonotope—a structure closed under Minkowski addition. This provides an inner approximation of the constraints, and this structure requires less memory than the original problem hence helping with the curse of dimensionality and providing a feasible (although suboptimal) solution. A zonotopic approximation is bottom-up: constraints associated with each asset in a VPP are approximated with a zonotope and then summed up with the Minkowski addition yielding the zonotope of the entire VPP. This enables us to efficiently add new assets into the pool (polynomial complexity) provided that the constraints associated with each asset are linear. In [37], the authors explore the management of virtual power plants and their alliances involving wind and solar parks. They propose a model involving the separate operation of assets but joint scheduling of them. Their model is based on Shapley value theory for profit distribution, and it is applied to case studies in China. In [6], the authors provide ADMM-based dispatch techniques for virtual power plants where they propose their own algorithm in which each DER communicates information only with its neighbors to collectively find the global optimal solution. The algorithm is fully distributed, it does not require a central controller, and the proof of its convergence is presented. In [17], the authors propose the algorithm of the implementation of ADMM within the blockchain while conducting the aggregation by means of a smart contract which can be used for any ADMM-based algorithm. The main point is that the aggregation relates to elementary operations with matrices and vectors i.e., the operations that can be implemented within ADMM in a matter of milliseconds. In [15], the authors provide the management of virtual power plants where gas-fired power plants balance a wind park, and in the wind forecast methods, they use spatial correlations. In [15], the authors propose

a technique called Robust Model Predictive Control which is also employed in the management of the VPP in this deliverable. In [35] and [36], the authors utilise metaheuristic algorithms such as Stochastic Fractal Search for the optimal reactive power flow of 118-bus systems. In [38], [39], [40], [41], the authors propose the management of virtual power plants by mixed-integer optimisation algorithms [44]. Enumeration of all optimisation methods would exceed the scope of this publication, but we can refer the reader to [28], which provides a thorough categorisation of these methods in the field and to [31], which provides a literature survey of unit commitment methods.

There are studies where VPPs include biogas power plants e.g., in [34], [42], and [43]. In [32] and [33], the VPP includes biogas power plants. The operation of this entity involves solving a large-scale mixed-integer linear programming problem (MILP). We also solve large-scale MILP problems. The focus is on balancing of wind parks, but it can easily be shown that the logic of the presented algorithms can be translated to broader classes of intermittent sources [30].

The main contribution of this research is a formulation of the *Gradual Increase* method which is a heuristic that was applied in the optimisation of pools of assets and is described in detail in [27], and this study provides further extensions of this methods.

Gradual Increase is a decomposition technique based on the operations of sub-pools of assets and with warm starts. The utilisation of this method enabled us to significantly accelerate the calculations and by up to two orders of magnitude (it was in the best case [27]). In general, this method demonstrates significant improvements in revenues compared to the brute force approach when limited by times typical for real-time applications, which is shown in Section 7. The *Gradual Increase* method is compared with other methods such as Proximal Jacobian ADMM in terms of average calculation time. We also demonstrate further improvements in average calculation times when a hybrid of *Gradual Increase* and Partial Integrality is employed. This deliverable provides further improvements in this method and demonstrates classes of problems that can be solved with the new formulation of the *Gradual Increase* method which the classical version of *Gradual Increase* cannot solve.

The method of *Gradual Increase* is tested within real-time optimisation settings: a MILP problem is solved every 24 h taking into account the state of the system (storage levels and states of the units) and the forecasts of prices and imbalances. The forecasts and the system's states are updated every 24 h and apart from the optimisation, we implement financial settlements associated with optimisation decisions. These settings are formulated by means of MPC and RMPC where the latter enables us to incorporate randomness into the model [15]. Large-scale MILP problems with four up to hundreds of assets are solved in the deterministic environment (MPC) using only commercial price forecasts; problems with the three assets are solved in two environments: MPC and RMPC, and three more forecasting methods are employed there.

1.1 Scope of this document

This document is the report on Task 4 of Work Package 3: "Optimisation of a VPP consisting of variable and dispatchable renewable energy sources".

The objective of this task is to analyse the possibilities of balancing intermittent electricity generation from variable renewable energy sources (wind and solar) with non-dispatchable renewable energy sources. The analysed assets are as follows: a battery, four biogas plants and a wind farm. The biogas plants and the battery are intended to balance the wind farm. In our framework, the operator of the wind farm nominates the schedule that can be produced, after which this schedule becomes mandatory for the system: the flexible assets will ensure that the nominated schedule is realised.

The biogas plants in our portfolio do not have typical timing constraints, nor operating costs, except for the condition that the total number of turbines switched per year should not exceed a predefined number. Besides examining these VPP (Table 7), we also consider hypothetical pools of biogas plants with hundreds of assets and with turbines with classical timing constraints to broaden the scope and applicability of the proposed optimisation methods.

Special attention is paid to day-ahead energy price forecasts and the quality of the forecasts is measured by the improvements in cumulative earnings.

Improvements in the forecasting performance of prices and in optimisation methods are important to allow a better rebalancing of variable renewable energy sources, thus promoting a further increase of renewable energy sources in the overall energy mix.

1.2 How to read this document

This report discusses the optimisation methods for maximising total revenue, focusing on two basic features: accuracy in the presence of uncertainty and the robustness of the optimisation methods to the size of the problem, i.e. the number of assets in the VPP. The detailed characteristics of the problem are given in the chapter Nuances of optimisation. Section 2 contains the description of the assets, the optimisation problem and the environment in which it takes place. Section 3 gives the structural form of the optimisation problem, which is suitable for asset-based decomposition and splitting. Section 3 also describes our method called *Gradual Increase*. Section 4 is devoted to Model Predictive Control techniques. Section 6 lists the specific features of the optimisation. Section 7 presents the results: Section 7.1 studies the robustness of the system against the increasing number of assets; Section 7.2 studies the accuracy of the different price predictions; and then we proceed to the conclusion.

2. Problem Description

This section first describes the environment in which the assets are optimised; then it describes the assets; and finally, it formulates the optimisation problem.

2.1 Power markets and the balancing task

All assets considered in this study operate in the EPEX SPOT market. Their aim is therefore to maximise profits from balancing wind farms. EPEX SPOT consists of two markets: the day-ahead (DA) market and the intraday (ID) market. The first is a market where a blind auction is held, and the price is the same for every participant. This is called the market clearing price. The price is locked in for the next day. The intraday market allows positions on the day-ahead market to be adjusted and short-term forecasts to be exploited. The market consists of two parts: the auction, which functions in the same way as the day-ahead auction, and a continuous market where trading takes place up to five minutes before delivery and where each participant is a price maker. The prices in the DA auction are not known in advance, so price forecasts are used when operating in the DA market. Intraday prices are known to be more volatile than DA prices, but they are centered around DA prices. In this study, the following price forecasts are used: commercial price forecasts; commercial price outlook; N-BEATS, DeepAR, mSSA. The same day-ahead prices are used for intraday prices. It is important to optimise against prices because it allows the owners of flexible assets to earn and they will otherwise not be willing to join a VPP if it is not profitable. Solar and wind power production is dependent on weather conditions and therefore flexible assets are needed to cover the imbalance. The purpose of the flexible assets is to maximise profit while covering the imbalance equal to: Realised Production - Predicted Production. The imbalance is difficult to predict, but the decisions made in the day-ahead market can be adjusted in the intraday market. In other words, the inaccuracy in the forecasts of imbalance can be mitigated by simple operations in the intraday market. There are two inputs into the optimisation:

- The vector of price forecasts on the DA market $Frc_t^{DA},\ t = 1,2,\dots,T$ where T is the prediction horizon.
- The vector of imbalance $IB_t,\ t = 1,2,\dots,T$ where T is the prediction horizon.
- If $Frc_t^{DA} < 0$, then $IB_t = IB_t^-$, i.e. it equal the negative part of the number IB_t. If the forecasted market price is below zero, it means that the grid is overloaded therefore additional injections are costly for both: the VPP and the grid. Such situations are handled in the ID market.

2.2 Features of the intraday and day-ahead markets

On the EPEX SPOT market, there are two trading systems: the system for managing blind auctions on the day ahead and intraday markets – EPEX Trading System (ETS), – and the system for the continuous intraday market, called M7. ETS is operated by the DeutscheBörse Group. It collects market orders in the limit order book (LOB). The sell orders from the LOB form the demand curve and the buy orders form the supply curve. The intersection of the demand and supply curves produces the market clearing price (MCP). This principle is applied to both auctions: the day ahead auction and the intraday auction, and in both cases the Euphemia algorithm is used. On the Continuous Intraday Market, the orders are collected in the order book and once the matching opposite order is found, the settlement takes place. Some orders are not executed on this market due to the lack of the matching counterpart orders. This order book is stored and managed by the M7 system. The following figure shows the structure of the EPEX SPOT market with an emphasis on the trading systems.

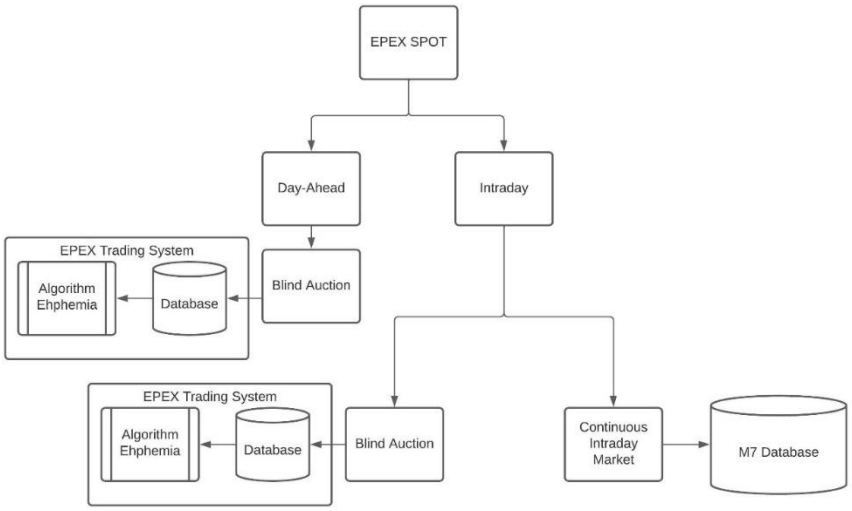

Figure 1 - The structure of the EPEX SPOT market

For small participants in the day-ahead market, it is possible to use price forecasts to nominate the volumes of electricity produced. This approach is used in this study. In general, it is possible to use optimisation to generate price-volume bids. In our simulations, we use ID-3 prices in the ID market for price estimation. As a subject of further research, we consider the use of the full order book stored in the M7 database for operations in the continuous ID market where decisions are made using optimisation.

2.3 Market Design

Given the presence of a large number of renewable energy sources, it is necessary to find ways of integrating them into the grid. There are two approaches:

- The increase in the share of flexible assets managed by a VPP that reacts quickly to undesirable events in the market using optimisation and forecasting techniques.
- Changes in the market design.

The former is the main subject of this study. The latter deserves to be a whole research project, but in the opinion of the authors, this research requires considerably expensive data sets. Major changes in market design are difficult to assess without testing them. For example, if a researcher proposes an alternative to the order book for a market design, the value of this idea can be assessed by the existence of a parallel market that would operate on that new design. The performance indicators of both markets can then be compared, and it will be possible to draw conclusions. An alternative to this parallel market is rigorous mathematical proof that a particular idea works, but this is very rare in this field. It is usually based on statistical hypotheses that work on things that have not been rejected on the basis of tests with predetermined assumptions, rather than working on things that have been proven.

An example of the test in market design changes is the introduction of flow-based market coupling. The effect is clear and the possibility of supplying the missing energy from another country reduces the risk of negative prices and blackouts. There is a lot of literature confirming these conclusions. Because of the positive effects of market coupling, the name of the XBID market was changed to SIDC - Single IntraDay Coupling. Similarly, important propositions in market design need to be tested. It is not enough to crunch numbers. Without large-scale testing platforms, these ideas are no more than opinions.

2.4 Description of the assets

There is a portfolio of four biogas power plants and a battery, and their goal is to balance the production of wind parks located in four regions of Germany: the power plants and the battery are

also located in Germany (Originally there were two power plants, but now there are four). Operators of the wind assets develop forecasts of how much energy their assets will produce, however the wind production is uncertain because it is weather dependent therefore balancing such assets is of utmost importance. The purpose of the pool of biogas plants and batteries is to address the cumulative imbalance. Note, the names of the biogas plants, the batteries, and the wind farms are not disclosed for reasons of confidentiality. Operational data for four biogas power plants and one battery was collected. The turbines within the biogas power plants have no typical timing constraints (minimum on and off time) except for the condition that the maximum number of switches per year is limited to 1500. When exploring the speed of our algorithms, timing constraints are imposed because they are typical for the turbines. For the pools of assets, the impact of scaling is tested in order to explore how the increasing number of assets affects the speed of the calculations: when scaling the pools of assets, the number of assets is predefined and the storage capacities, gas inflows, maximum and minimum productions of the turbines and minimum on and off times for the turbines within biogas power plants (timing constraints) are randomly assigned. In the scaling of the pools of flexible assets, the size of wind parks increases proportionally. This enables us to apply the proposed algorithms for a broader scope of asset types. We operate on the German market EPEX SPOT (day-ahead and intra-day) and use the price forecasts as objective value coefficients. The imbalance is modeled by means of autoregressive models. In our biogas power plants the processes of fertiliser production and electricity production are separated. The operators of the assets are guaranteed to obtain fixed amounts F_k (see Table 1) of biogas (measured in MWh) in every period and they concentrate only on electricity production. In this study, biogas power plants with heating rods (the usage of heat as a by-product of electricity production) are not considered.

2.5 The Notation

The formulation of the optimisation models requires a specific notation which is provided in Table 1 and Table 2.

Table 1 - Notation: constants

The Symbol	Explanation
$\#A$	the number of assets
$\#U(k)$	the number of turbines within asset k
E_k^{max}	the maximum storage of asset k (MWh)
F_k	the flow of biogas power plant k every 15 minutes (MWh)
η_k^d	the efficiency of the discharge of battery k
η_k^c	the efficiency of the charge of battery k
$Pmin_{ki}$	the minimum power of Turbine i of Asset k (MW)
$Pmax_{ki}$	the maximum power of Turbine i of Asset k (MW)
$Frc_t^{(DA)}$	the day-ahead price at time t
C_{ki}^w	the cost of switching-on of Turbine i of Asset k
C_{ki}^v	the cost of switching-off of Turbine i of Asset k at time t
$UT^{(k,i)}$	the minimum on time of Turbine i of Asset k at
$DT^{(k,i)}$	the minimum off time of Turbine i of Asset k at
$\forall t, k, i$	the shorthand for: for all $t \in \{1, \dots, T\}$ and all $k \in \{1, \dots, \#A\}$ and all $i \in \{1, \dots, \#T(k)\}$

Table 2 - Notation: Variables

The Symbol	Explanation
$p_{t,k,i}$	the total power produced at time t by Turbine i of Asset k
$u_{t,k,i}$	the state of Turbine i of Asset k at time t
$v_{t,k,i}$	the switch-on decision of Turbine i of Asset k at time t
$w_{t,k,i}$	the switch-off decision of Turbine i of Asset k at time t
$soc_{t,k}^{Bg}$	the storage level of Asset (Power Plant) k at time t (MWh)
$soc_{t,k}^{Bt}$	the storage level of Asset (Battery) k at time t (MWh)
$p_{t,k,1}^{d}$	the discharge of battery k at time t
$p_{t,k,1}^{c}$	the charge of battery k at time t
$TotalPower_t$	the total power produced by all assets at time t
$TotalVcost_t$	the total switch-on costs produced by all assets at time t
$TotalWcost_t$	the total switch-off costs produced by all assets at time t
x_k	the set of all variables p, u, v, w, soc, p reduced to asset k
y_t	the set of all variables p, u, v, w, soc, p reduced to time t
z^*	the optimal value of variables z. Any variable superscripted with a star denotes the value optimal for the objective function

2.6 The Objective

The objective is to maximise the profits of pools of biogas power plants and batteries while balancing the wind park and taking technical constraints into account. The technical constraints of the turbines are their maximum and minimum production per period, their timing constraints, and the storage levels of the fuel.

The objective to be maximised is as follows:

$$\sum_{t=1}^{T} Frc_t^{(DA)} \cdot TotalPower_t - \sum_{t=1}^{T} TotalVcost_t - \sum_{t=1}^{T} TotalWcost_t, \quad (1)$$

where $Frc_t^{(DA)}$ is the price forecast on the day ahead market, and the variables in the objective are represented by the following equations (see Table 2):

$$TotalPower_t = \sum_{k=1}^{\#A} \sum_{i=1}^{\#U(k)} p_{t,k,i}, \quad TotalVcost_t = \sum_{k=1}^{\#A} \sum_{i=1}^{\#U(k)} C_{ki}^{v} v_{t,k,i}, \quad TotalWcost_t = \sum_{k=1}^{\#A} \sum_{i=1}^{\#U(k)} C_{ki}^{w} w_{t,k,i}. \quad (2)$$

The revenues from the sale of energy on the market are optimised penalising every switch-on and -off with costs C_{ki}^{v} and C_{ki}^{w}, where k denotes the index of an asset and i is the index of a turbine. In the case of a battery, these costs are zero.

The constant T denotes the prediction horizon that is used for the optimisation and $Frc_t^{(DA)}$ is the forecasts of day-ahead prices at time t.

2.7 The Constraints

In this subsection, we describe all of the constraints associated with biogas power plants.

2.7.1 Binary Constraints

We employ 3bin formulation [16], and all u, v, and w variables (Table 2) are binary, i.e.

$$u_{\{t,k,i\}}, v_{\{t,k,i\}}, w_{\{t,k,i\}} \in \{0,1\} \quad \forall\, i,k,t. \tag{3}$$

2.7.2 Constraints for the biogas power plants

In this subsection, we write out all the constraints associated with biogas power plants.

2.7.2.1 Power Constraints

For every turbine we have two basic values: $Pmin_{ki} > 0$ and $Pmax_{ki} > Pmin_{ki}$ which implies that it can either do nothing or produce within the interval $[Pmin_{ki}, Pmiax_{ki}]$ i.e.

$$p_{t,k,i} \in \{0\} \cup [Pmin_{ki}, Pmax_{ki}] \quad \forall\, t,k,i$$

These constraints can be written using the $u_{t,k,i}$ variables which equal 1 if at time t the i-th turbine of Asset k is on and it is 0 otherwise:

$$p_{t,k,i} - Pmin_{ki}\, u_{t,k,i} \geq 0, \qquad Pmax_{ki}\, u_{t,k,i} - p_{t,k,i} \geq 0 \quad \forall\, t,k,i. \tag{4}$$

2.7.2.2 Storage constraints

Every biogas power plant k has its own storage and a constant flow of gas F_k (expressed in MWh) into it. The new storage level is equal to the old level added by F_k and subtracted by the aggregate energy produced by all turbines within Asset k during one period:

$$soc_{t,k}^{Bg} = soc_{t-1,k}^{Bg} + F_k - \sum_{i=1}^{\#U(k)} p_{t,k,i} \cdot \Delta\, t \quad \forall\, t,k, \qquad \Delta\,\mathrm{t} = 15\ \text{minutes} \tag{5}$$

and we have the box constraints:

$$soc_{t,k}^{Bg} \in [0, E_k^{max}] \quad \forall t,k. \tag{6}$$

2.7.2.3 Timing constraints

The on and off decisions for the turbines are conducted via binary switch-on (v) and binary switch-off (w) variables as follows [16]:

$$u_{t,k,i} - u_{t-1,k,i} = v_{t,k,i} - w_{t,k,i} \quad \forall\, t,k,i. \tag{7}$$

If at time t a turbine i of asset k is on or off than it has to be on or off for at least $UT^{(k,i)}$ and $DT^{(k,i)}$ periods, respectively, which is exressed as follows:

$$\sum_{j=t-UT^{(k,i)}+1}^{t} v_{j,k,i} \leq u_{\{t,k,i\}} \text{ and } \sum_{t-DT^{(k,i)}+1}^{t} w_{j,k,i} \leq 1 - u_{t,k,i} \quad \forall\, t,k,i. \tag{8}$$

2.7.2.4 Timing constraints tuning

There are light requirements on our turbines of the biogas power plants: maximum number of switches per year. These conditions can be handled by imposing penalties for switch-ons and switch-offs. On the other hand there are, generally, turbines with strict timing constraints therefore in our exploration of the speed of the decomposition algorithms, timing constraints are intentionally imposed in order to broaden the scope of applications of our algorithms. In any case, if it is possible to handle the timing constraints by penalisations C^v and C^w, we should do this because the constraints (9) account for most of inequality constraints of the problems and their removal can lead to significant reductions of the computation time.

2.7.3 The constraints for batteries

In this subsection, we write out all the constraints associated with batteries.

2.7.3.1 Power constraints

For every battery we have the following power constraints:

$$\begin{aligned} Pmax_{k,1}\, u_{t,k,1} - p^d_{t,k,1} &\geq 0 \quad \forall\, t,k, \\ Pmax_{k,1}\,(1-u_{t,k,1}) - p^c_{t,k,1} &\geq 0 \quad \forall\, t,k, \\ p^c_{t,k,1} - Pmin_{k,1}\, w_{t,k,1} &\geq 0 \quad \forall\, t,k, \\ p^d_{t,k,1} - Pmin_{k,1}\, v_{t,k,1} &\geq 0 \quad \forall\, t,k. \end{aligned} \tag{9}$$

Constraints (10) ensure that we cannot charge and discharge at the same time. The update of the storage level and the state of charge is conducted as follows:

$$soc^{Bt}_{t+\Delta\, t,k} = soc^{Bt}_{t,k} + \Delta t \cdot \eta^c_k \cdot p^c_{t,k,1} - \frac{p^d_{t,k,1}\Delta\, t}{\eta^d_k} \quad \forall\, t,k, \text{ with } p^d_{t,k,1} \text{ constant in } [t, t+\Delta t] \text{ and } \Delta t \text{ is always 15 minutes.} \tag{10}$$

2.7.3.2 Tightening constraints

$$v_{t,k,1} \leq u_{t,k,1} \text{ and } w_{t,k,1} \leq 1 - u_{t,k,1} \quad \forall\ t,k, \tag{11}$$

and for all t we have box constraints:

$$soc^{Bt}_{t,k} \in [0, E^{max}_k] \ \forall\, t,k. \tag{12}$$

2.7.3.3 Switches of the batteries

The on and off decisions for the batteries are conducted via binary switch-on (v) and binary switch-off (w) variables as follows:

$$u_{t,k,1} - u_{t-1,k,1} = v_{t,k,1} - w_{t,k,1} \quad \forall\, t,k. \tag{13}$$

2.7.3.4 Components of the objective function

We define the variables $p_{t,k,1}$ as follows:

$$p_{t,k,1} = p^d_{t,k,1} - p^c_{t,k,1} \quad \forall\, t,k, \tag{14}$$

which is included in the objective (1).

2.7.4 Coupling constraints

The coupling constraint is a task that all flexible assets must fulfill. This is given by the imbalance $IB_t \ \forall\, t$ between the realised and forecasted production of the intermittent sources:

$$\sum_{k=1}^{\#A} \sum_{i=1}^{\#U(k)} p_{t,k,i} \ \geq \ IB_t\,, \qquad \forall\, t. \tag{15}$$

Let us note that the right-hand side of (16) is the input that we must estimate. This boils down to estimating the imbalance which is a challenging task. In this study we approach the imbalance by means of two methods:

- Taking the historical imbalance: perfect foresight. We will use it as a benchmark. This will be the upper bound to the problem.
- Fitting the imbalance with ARMA processes and using simulations.

2.7.5 Optimisation problem formulation

Summarising the constraints and objectives, we can formulate the optimisation problem with the forecast parameters: Frc^{DA}, Frc^{ID}, and $Imbalance$ as follows:

$$\text{maximise} \sum_{t=1}^{T} Frc_t^{(DA)} \cdot TotalPower_t - \sum_{t=1}^{T} TotalVcost_t - \sum_{t=1}^{T} TotalWcost_t$$

$$s.t. \qquad (2) - (15).$$

which is denoted as follows: $VPP(Frc^{DA}, IB)$. For the problem $VPP(Frc^{DA}, IB)$, we define the additional notation:

- $VPP_x(Frc^{DA}, IB)$ is the optimal solutions of the problem, namely:

$$VPP_x(Frc^{DA}, IB) = (\,p^*, p^{c*}, p^{d*}, u^*, v^*, w^*, soc^*)$$

- $VPP_r(Frc^{DA}, IB)$ is the realised value of the problem, i.e. $VPP_r(Frc^{DA}, IB) =$

$$= \sum_{t=1}^{T} Frc_t^{(DA)} \cdot TotalPower_t^* - \sum_{t=1}^{T} TotalVcost_t^* - \sum_{t=1}^{T} TotalWcost_t^*$$

- $VPP_{rv}(Frc^{DA}, IB)$ is the revenue yielded from $VPP_x(Frc^{DA}, IB)$ i.e.

$$VPP_{rv}(Frc^{DA}, IB) = \sum_{t=1}^{T} Frc_t^{(DA)} \cdot TotalPower_t^*$$

  which is equivalent to

$$VPP_{rv}(Frc^{DA}, IB) = VPP_{rv}(Frc^{DA}, Frc^{DA}, IB) =$$
$$= \sum_{t=1}^{T} Frc_t^{(DA)} \cdot (TotalPower_t^* - Task_t) + Frc_t^{(DA)} \cdot Task_t -$$
$$- \text{penalty for violation coupling constraint}$$

  Because $Frc^{DA} = Frc^{ID}$. However, when the realisations of the ID and DA prices are different. This is the realised revenue when the prices and the imbalance become known. When we the forecast of the imbalance IB is replaced with its realisation IB^R, then it is checked whether the feasibility is preserved. If so the sum $VPP_{rv}(Spt^{DA}, Spt^{ID}, IB)$ will be

directly applied as the realised value. Otherwise, the missing energy is bought on the market.

- $VPP_{rv}^{(h)}(Spt^{DA}, Spt^{ID}, IB^R)$ is the realised revenue when the prices and the imbalance become known within the execution horizon h, i.e.

$$VPP_{rv}^{(h)}(Spt^{DA}, Spt^{ID}, IB^R) = \sum_{t=1}^{h} Spt_t^{(DA)} \cdot (TotalPower_t^* - Task_t) + Spt_t^{(ID)} \cdot Task_t$$

  This value will be used in *Model Predictive Control*.

- $VPP_{rv}^*(Spt^{DA}, Spt^{ID}, IB^R)$ is the realised revenue from the solution of the perfect foresight problem $VPP(Spt^{DA}, Spt^{ID}, IB^R)$, where IB^R is the realised imbalance. The settings of Problem $VPP_{rv}^*(Spt^{DA}, Spt^{ID}, IB^R)$ correspond to the state of perfect foresight, i.e. when all the uncertain values are known up front. This value is used as the benchmark.

- Our goal is to elaborate such a strategy that the ratio

$$\sigma = \frac{VPV_{rv}(Spt^{DA}, Spt^{ID}, IB^R)}{VPP_{rv}^*(Spt^{DA}, Spt^{ID}, IB^R)}$$

  is as close to 1 as possible which can be increased by better optimisations and better forecasts.

We also introduce the optimisation problem $VPP(Frc^{DA}, Frc^{ID}, IB, State)$ which differs from $VPP(Frc^{DA}, Frc^{ID}, IB)$ by specifying starting conditions of turbines and storage levels constrained in the variable $State$.

3. Gradual Increase in Structural Asset-Based Problem Formulation

The problem $VPP(Frc^{DA}, IB)$ has a linear objective and all linear constraints, therefore it can easily be shown that it can be rewritten in the following form:

$$\text{maximise} \sum_{k=1}^{\#A} p_k^T \cdot x_k \tag{16}$$

$$\text{s.t} \quad \sum_{k=1}^{\#A} A_k^T \cdot x_k \geq a, \tag{17}$$

$$B_k \cdot x_k \leq b_k \tag{18}$$

$$x_k(I) \in \{0,1\}, \forall\, k \in \{1,2,3,\dots,\#A\} \tag{19}$$

where x_k represents all variables associated with Asset k, in other words x_k constrains all variables from Table 2, whose Asset's index is k. The vector p_k consists of all objective value coefficients from Expression (1) associated with Asset k and Expression (16) is equivalent to Expression (1). In an analogous manner Equation (17) is equivalent to Equation (15) and Expression (18) is equivalent to Expressions (2),(4)—(14). Expression (19) is equivalent to (3).

3.1 Description of gradual increase

The main idea behind the method of *Gradual Increase* lies in the proper use of the *warm start*. When we dealing with the task a from (18), one can check whether it is possible to implement it with a smaller number of assets (or turbines within the assets). If it is possible, then the resulting solution can be used as a start in either a larger or the entire pool. We try to start with a minimum sub-pool capable of implementing the task and add assets to the pool with its consequent optimisation, until the entire pool is achieved. When implementing this algorithm, it is important to ensure that the *branch and bound* trees will not be destroyed which is achieved by the parallel run of the problem containing all assets which is interrupted when new feasible solutions are found: the problem is fed with that new feasible solution and then, the optimisation is resumed. This can be achieved by the usage of built-in callback functions within GUROBI are so complex that even the best *solver* will not be able to find a feasible solution to it. However, *Gradual Increase* enables us to find a feasible solution for the entire pool from the solution of subproblems. It involves systems of subsets of the pool of assets and three kinds of sub-problems:

- **Initial problem** – a smaller problem with fewer assets, but the same right-hand side of the coupling constraint.
- **Independent problems** – maximisation of revenues of each asset independently and without the coupling constraint.
- **Intermediary problems** – problems taking the warm starts from previous sub-pools of assets.

3.1.1 Systems of subsets

Let $\#A$ denote the number of the assets in the entire pool. Then any sub-pool is j belongs to the system of subsets of $2^{\{1,2,\dots,\#A\}}$, i.e. $SubSet(j) \subset 2^{\{1,2,\dots,\#A\}}\ \forall j \in \{1,2,\dots,\#A\}$. The initial sub-pool is denoted $SubSet(0)$, and $\#SubSet(i) > \#SubSet(j)$ whenever $i > j$. The method is called *Gradual Increase* because of the gradual increase of the sizes of the sub-pools involved in the implementing the task in the coupling constraint.

3.1.2 The initial problem

The initial problem is a reduction of Problem (16)—(19) reduced to the first subset $SubSet(0)$, i.e.:

Initial Problem:

$$\text{maximise} \sum_{k \in SubSet(0)} p_k^T \cdot x_k \tag{20}$$

$$\text{s.t.} \sum_{k \in SubSet(0)} A_k^T \cdot x_k \geq b, \tag{21}$$

$$B_k \cdot x_k \leq b_k, \tag{22}$$

$$x_k(I) \quad \in \{0,1\}, \tag{23}$$

$$\forall\, k \in SubSet(0). \tag{24}$$

Since $\#SubSet(0) < \#A$, the Problem (20)-(24) contains fewer variables and constraints than the initial problem and should be solved faster except for specific cases, e.g. when the pool is so small that it is overloaded. The solution of this problem yields a sequence of vectors $z_k^{(0)}$, $k \in SubSet(0)$.

3.1.3 The independent and intermediary problems

Independent problems refer to cases where each asset is optimised independently with no coupling constraint, i.e. there is only one aim: maximisation of the revenue.

The initial problem and independent problems require no additional inputs while the intermediary problems take the output of both to build a warm start. If an intermediary problem has the sub-pool $SubSet(j+1)$ its warm start is:

$$x_m.start = \begin{cases} z_m^{(j)}, & m \in SubSet(j) \\ x_m^*, & m \notin SubSet(j) \end{cases} \text{ for } \forall\, m \in \{1,2,\dots,\#A\}$$

Table 3 formulates both problems:

Table 3 - Problems solved within GI

Independent Problems	Intermediary Problems
parfor $m \in \{1,2,\dots,\#A\}$ **do** maximise $p_m^T \cdot x_m$ s.t. $A_m^T \cdot x_m \geq \mathbf{0}$, $x_m(I) \in \{0,1\}$, $B_m \cdot x_m \leq b_m$. **end parfor** The optimal solutions of the independent problems are denoted as follows: $x_m^*, \quad m \in \{1,2,\dots,\#A\}$ In the diagrams there is a slightly different notation. E.g. x_3^* is denoted X3* in the diagrams. These inputs are the same for all sub-pools and can obtain in parallel. Let us call vectors x_m^*, $m \in \{1,2,\dots,\#A\}$ independent schedules.	maximise $\sum_{k \in SubSet(j+1)} p_k^T \cdot x_k$ s.t. $\sum_{k \in SubSet(j+1)} A_k^T \cdot x_k \geq a$, $B_k \cdot x_k \leq b_k, \quad x_k(I) \in \{0,1\}$, $\forall k \in SubSet(j+1)$, $x_m.start = z_m^{(j)} \quad \forall m \in SubSet(j)$, $x_m.start = x_m^* \quad \forall m \notin SubSet(j)$. The solution of the intermediary problem with sub-pool $SubSet(j+1)$ is denoted $z_m^{(j+1)}$, $\forall m \in SubSet(j+1)$. Let us note that the combination of these variables with the independent schedules yields a feasible solution for the entire pool because if $\gamma_k = \begin{cases} z_k^{(j+1)}, k \in SubSet(j+1) \\ x_k^*, \quad k \notin SubSet(j+1) \end{cases}$ then $\sum_{k=1}^{\#A} A_k^T \cdot \gamma_k = \sum_{k \in SubSet(j+1)} A_k^T \cdot \gamma_k + \sum_{k \notin SubSet(j+1)} A_k^T \cdot \gamma_k =$ $\sum_{k \in SubSet(j+1)} A_k^T \cdot z_k^{(j+1)} + \sum_{k \notin SubSet(j+1)} A_k^T \cdot x_k^* \geq a.$

3.1.4 Pseudocode for Gradual Increase

What precedes the *Gradual Increase* is the choice of its subsets:

$$SubSet(0) \subset SubSet(1) \subset \cdots \subset SubSet(M)$$

$$\#SubSet(0) < \#SubSet(1) < \cdots < \#SubSet(M)$$

For this system of sub-sets, the following pseudocode is formulated:

Table 4 - The Pseudocode for GI

Gradual Increase Pseudocode
1. Solve in parallel all independent problems and the initial problem and store: $x_k^*, k \in \{1,2,\dots,\#A\}$ and $z_k^{(0)}, k \in SubSet(0)$. 2. **for** $j \in \{1,2,\dots,M\}$ **do** • Solve Intermediary Problem for $SubSet(j)$. • Store $z_k^{(j)}, k \in SubSet(j)$ • Produce feasible solution for the entire pool: $\gamma_k^{(j)} = \begin{cases} z_k^{(j)}, k \in SubSet(j) \\ x_k^*, \ k \notin SubSet(j) \end{cases}$' • for all $k \in \{1,2,\dots,\#A\}$ **end for** 3. The final feasible solution is $\gamma_k^{(M)}, k \in \{1,2,\dots,\#A\}$.

Figure 2 to Figure 7 demonstrate the technique of *Gradual Increase* for a pool with seven assets for different systems of sub-pools. The procurement of the independent schedules is visualised in the following diagram (seven assets):

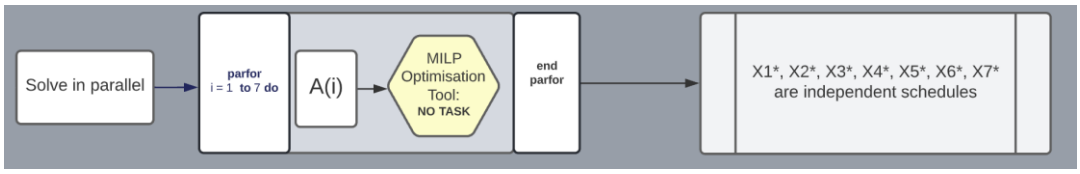

Figure 2 - Procurement of independent schedules by parallel runs.

The schedules X1*, X2*, …, X6*, X7* yield the largest optimal value but they may not provide a feasible solution. However, if some sub-pool, e.g. [1,2,3] produced a solution [X1,X2,X3] that satisfies the coupling constraint, then [X1, X2, X3, X4*, X5*, X6*, X7*] is a feasible solution for the entire pool.

The visualisation of the initial problem with $SubSet(0) = [1,3,7]$ is demonstrated in Figure 3. The following problem is equivalent to the scheme in Figure 3:

- The right-hand side of the coupling constraint, say a, is denoted as **Initial Task** in the diagram.
- In the initial problem, there is no warm start as is seen in the following problem formulation and the scheme.

Problem Formulation:

$$\text{maximise} \sum_{k\in\{1,3,7\}} p_k^T \cdot x_k$$

$$\text{s.t.} \sum_{k\in\{1,3,7\}} A_k^T \cdot x_k \geq a,$$

$$B_k \cdot x_k \leq b_k, \quad x_k(I) \in \{0,1\}, \forall k \in \{1,3,7\}$$

Equivalent Schematic Formulation:

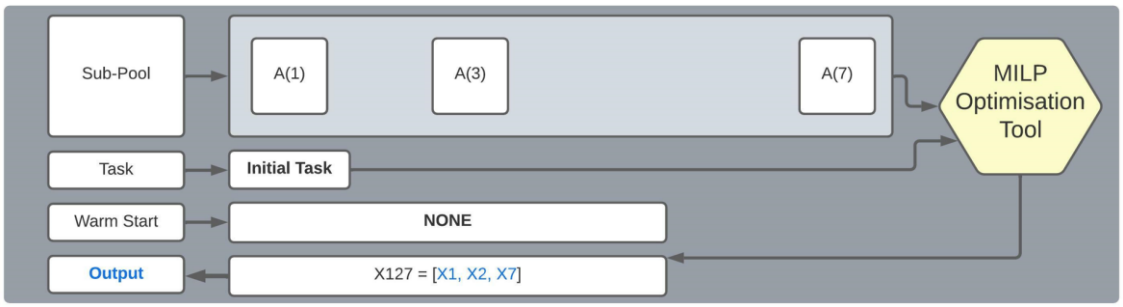

Figure 3 - Schematic visualisation of the initial problem for $SubSet(0) = [1, 3, 7]$.

The following diagram (Figure 4) demonstrates the problem where the entire pool is solved relying on only the power of the *solver*: no warm starts. The initial problem has a similar structure to this problem.

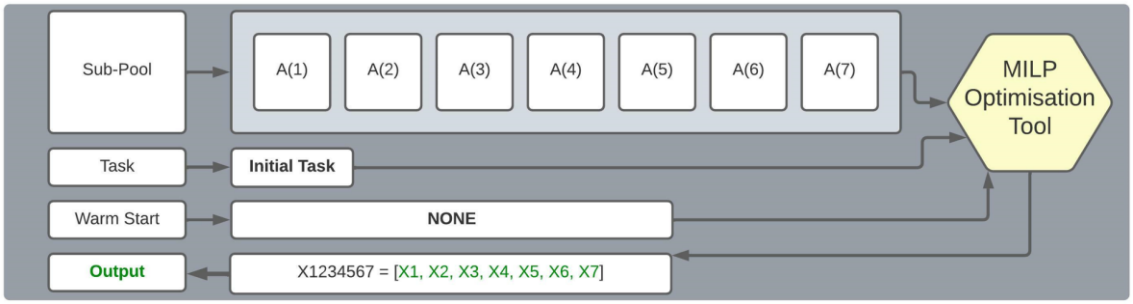

Figure 4 - Schematic visualisation of the initial problem which is solved without Gradual Increase

The intermediary problems in the diagram differ from the initial problem by the presence of warm starts. Figure 5 demonstrates the methodology of *Gradual Increase* on a system of sub-pools: $SubSet(0) = [1,2,3], SubSet(1) = [1,2,3,4,5], SubSet(2) = [1,2,3,4,5,6,7]$. The initial problem and the independent problems can be solved in parallel. Then the output of these problems can be concatenated and produce a warm start for the first intermediate problem. If some subset $SubSet(i)$ does not contain all assets, then the feasible solution is built by concatenating the corresponding solution with the independent schedules. In the following diagrams the part where independent schedules are calculated is removed but it is meant that it was calculated beforehand. In Figure 5, each sub-pool generates a feasible solution for the entire pool (see the right-hand side of the diagram) and the warm start for the larger pool. Figure 6 demonstrates the solution of a problem with seven assets by using another system of sub-pools: $SubSet(0) = [2,4]$, $SubSet(1) = [1,2,4]$, $SubSet(2) = [1,2,3,4,5], SubSet(3) = [1,2,3,4,5,6,7]$.

Different systems of pools can be handled in parallel. We call these systems shapes, because every pool can be visualised by a shape like in Figure 8.

Figure 7 demonstrates a specific case where the final pool is smaller than the entire pool. The feasible solution is built by concatenating with independent schedules. This specific case demonstrates that the generic *Gradual Increase* is about generating feasible solutions (lower bounds), but not the upper bounds which confirm the quality of the solution, and which are used to calculate the MIPgap. In order to handle this issue, the Tetris Notation and the corresponding insights were introduced.

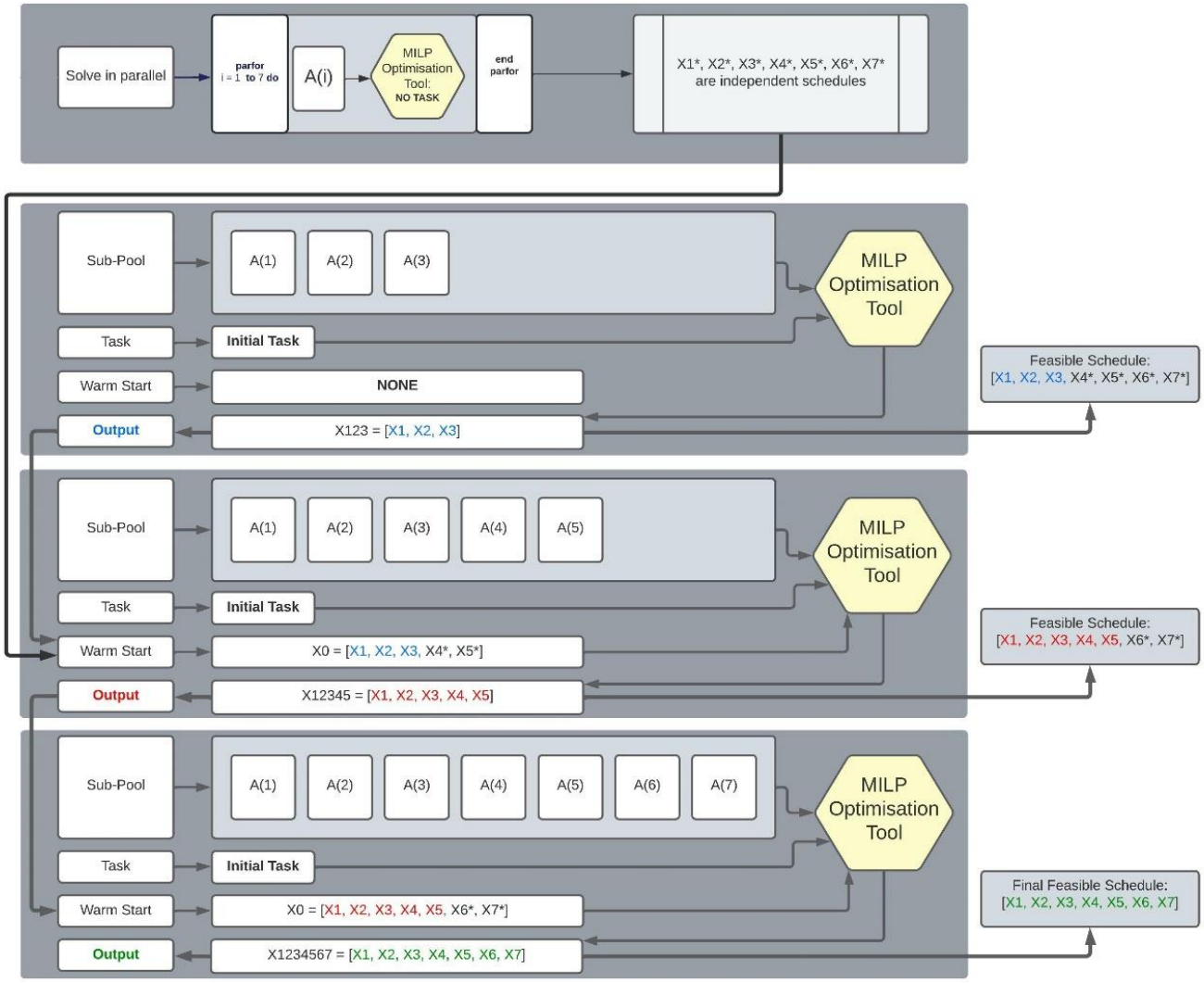

Figure 5 - Schematic visualisation of Gradual Increase for a pool with seven assets and the following system of subsets: $\boldsymbol{SubSet(0) = [1, 2, 3]}$, $\boldsymbol{SubSet(1) = [1, 2, 3, 4, 5]}$, $\boldsymbol{SubSet(2) = [1, 2, 3, 4, 5, 6, 7]}$.

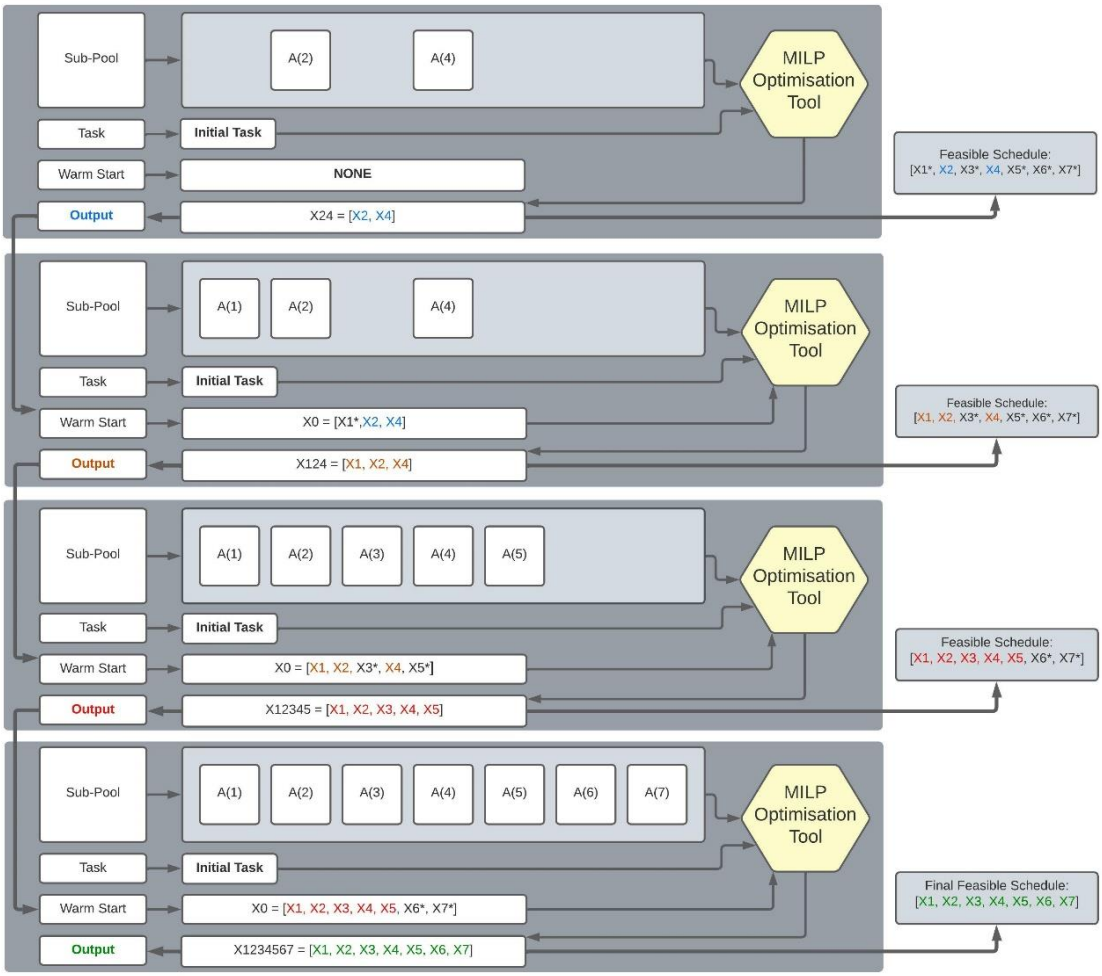

Figure 6 - Schematic visualisation of Gradual Increase for a pool with seven assets and the following system of subsets: $\mathbf{SubSet}(\mathbf{0}) = [\mathbf{2}, \mathbf{4}], \mathbf{SubSet}(\mathbf{1}) = [\mathbf{1}, \mathbf{2}, \mathbf{4}], \mathbf{SubSet}(\mathbf{2}) = [\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}], \mathbf{SubSet}(\mathbf{3}) = [\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}]$.

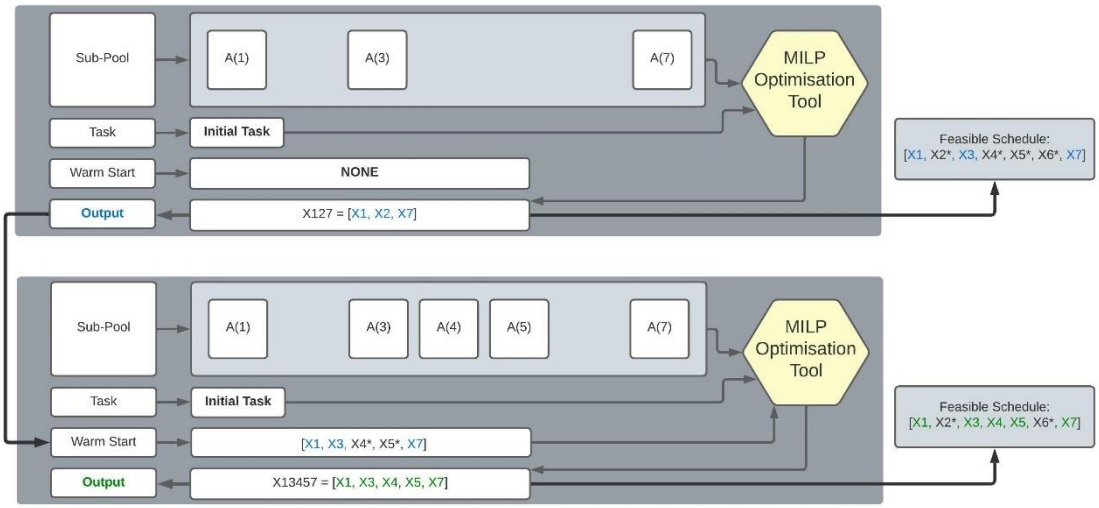

Figure 7 - Schematic visualisation of Gradual Increase for a pool with seven assets and the following system of subsets: $\boldsymbol{SubSet(0) = [1, 3, 7]}$, $\boldsymbol{SubSet(1) = [1, 3, 4, 5, 7]}$.

3.1.5 Tetris Notation

Figure 8 visualises different systems of subsets. For example, the top-left system of subsets was used in Figure 5, and the top-right system was used in Figure 6. This notation is called Tetris, because it resembles shapes in the Tetris game. The cells are numbered in Figure 8 to make analogies with Figure 5 and 6 clearer. In further diagrams the numbering is removed.

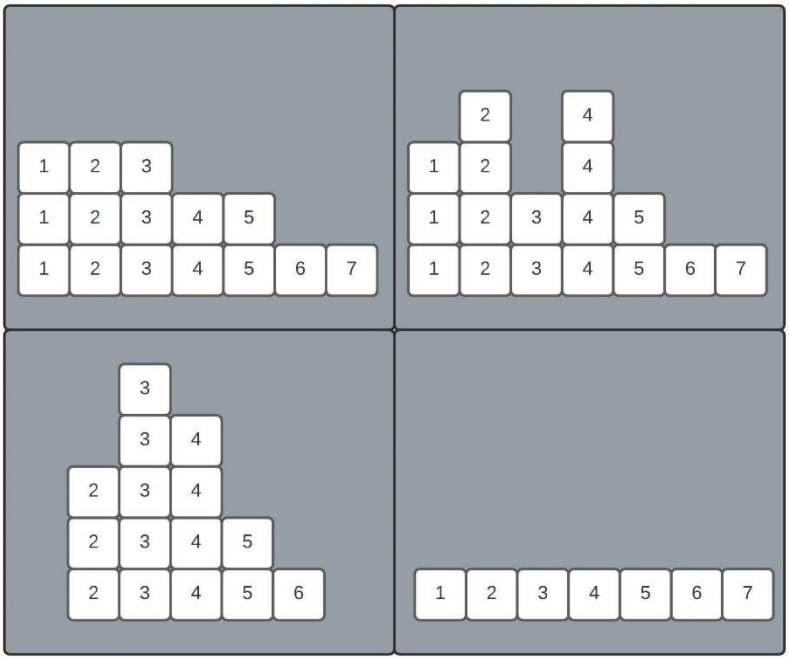

Figure 8 - Visualisation of different systems of sub-pools.

The bottom-left and bottom-right parts of Figure 8 are specific cases of systems of sub-pools. The bottom-left system does not involve all assets and following the notation from Figures 4 and 5, its final feasible solution is in the following form: [X1*, X2, X3, X4, X5, X6, X7*]. The schedule for assets 2 to 6 is obtained by optimising the pool of five assets and the final schedule is obtained concatenating that solution with X1* and X7*. This happens when the complexity of some assets is so high that presolve and processing of warm starts will require too much time which might exceed the time limit. In practical implementations this situation might happen when stopping criteria are minimum MIPgaps without time limits.

The system of sub-pools in the bottom left part of Figure 8 does not produce the upper bound of the entire pool. The only upper bound available is: $\sum_{k=1}^{\#A} p_k^T \cdot x_k^*$ which strongly depends on the size of the task.

The system of sub-pools in the bottom-right part of Figure 8 is a trivial case where no *Gradual Increase* takes place and Problem (16)—(19) is solved from scratch. This system might not produce a feasible solution, but in more than 80% of cases its upper bound is the highest. The upper bounds are needed to confirm that the correspond feasible solution (lower bound) satisfies predefined criteria. Therefore, it is recommended to run in parallel to *Gradual Increase* this trivial pool and then to take the largest lower bound and the smallest upper bound.

Figure 8 demonstrates the shorthand notation analogous to the notation in Figure 7, but it also contains two more components: the hexagon and the frame. The former denotes a computational tool used for optimising a predefined shape within *Gradual Increase*; the latter denotes vectors of independent schedules that are used in concatenating for obtaining feasible solutions and warm starts.

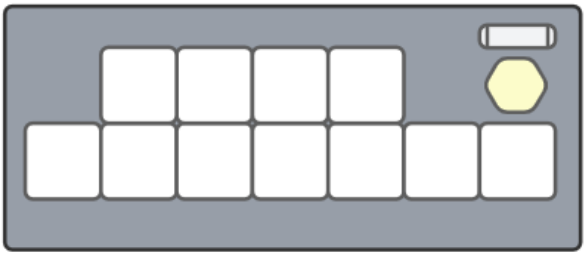

Figure 9 - Schematic visualisation. The hexagon represents the optimisation tool reserved for this shape. The frame above the hexagon is the set of independent schedules that are used for building warm starts.

3.1.6 Injection of feasible solutions

GUROBI enables the injection of a feasible solution to the solution process, but this can be computationally costly. This also relates to the warm starts. This process can be time consuming because of the need to integrate a feasible solution as a node of the Branch & Bound trees of the solution process. Figure 10 demonstrates an example when a feasible solution was processed in 26.14 seconds. This processing time can be even larger therefore the number of injected feasible solutions should be reasonable and tailored to the specific features of optimisation problems.

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Gurobi Compute Server Worker version 9.1.2 build v9.1.2rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 326100 rows, 298700 columns and 6985008 nonzeros
Model fingerprint: 0xbd594ae4
Variable types: 240400 continuous, 58300 integer (58300 binary)
Coefficient statistics:
  Matrix range     [5e-03, 6e+06]
  Objective range  [2e+00, 1e+04]
  Bounds range     [1e+00, 1e+08]
  RHS range        [1e+00, 6e+06]

User MIP start produced solution with objective 4.89335e+08 (8.84s)
User MIP start produced solution with objective 4.89579e+08 (18.55s)
User MIP start produced solution with objective 4.91813e+08 (20.28s)
User MIP start produced solution with objective 4.9182e+08 (20.36s)
Loaded user MIP start with objective 4.9182e+08
Processed MIP start in 26.14 seconds

Presolve removed 8455 rows and 9432 columns (presolve time = 5s) ...
Presolve removed 11276 rows and 10695 columns (presolve time = 10s) ...
Presolve removed 11241 rows and 10660 columns
Presolve time: 10.85s
Presolved: 314859 rows, 288040 columns, 1988932 nonzeros
Variable types: 229433 continuous, 58607 integer (58607 binary)
```

Figure 10 - A case when the processing of a warm start is time consuming – outlier. In our problems it takes on average 1-2 seconds to process warm starts.

Figure 11 describes the situation when five different *shapes* are solved in parallel. The upper part of Figure 11 represents the calculation of the independent schedules which are then used in all shapes except for the specific shape that includes all the assets.

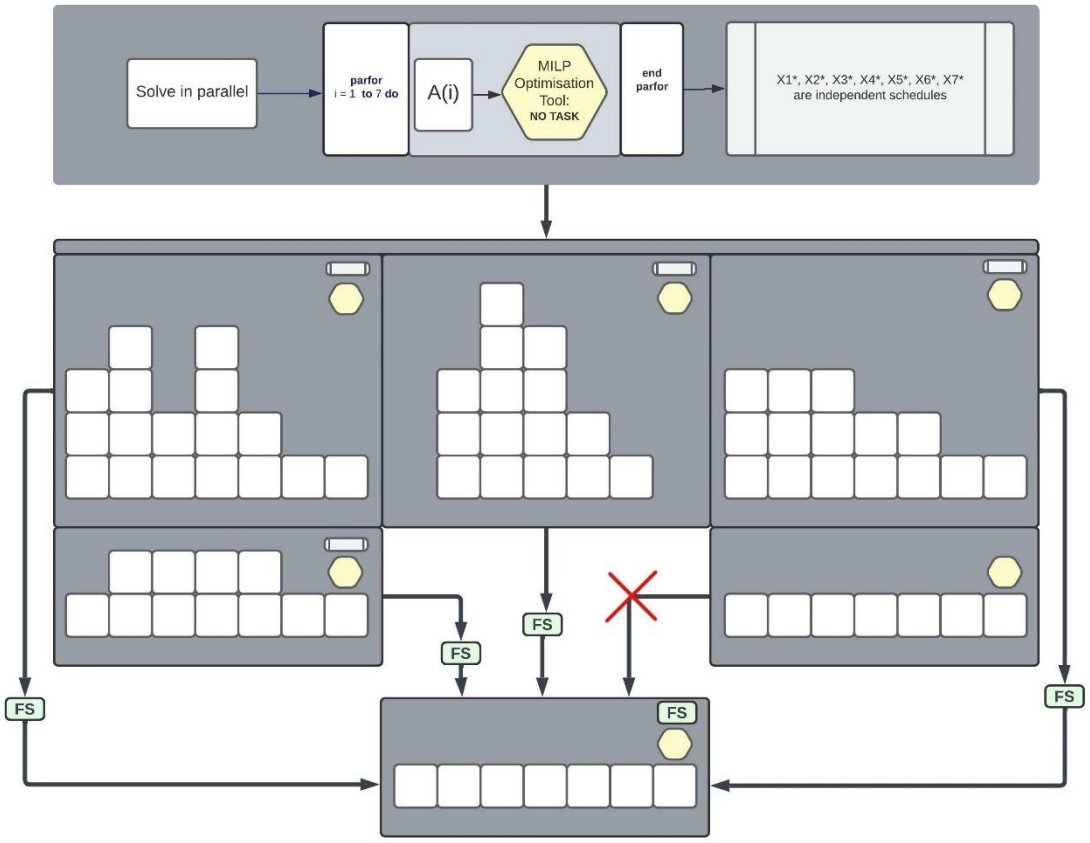

Figure 11 - Schematic visualisation of parallel handling of different *shapes*. The hexagon represents the optimisation tool reserved for a single *shape*. The frame above the hexagon is the set of independent schedules that are used for building warm starts.

In Figure 11, there are two shapes containing all assets: the bottom-middle shape takes get injections of the feasible solutions when every sub-pools of assets of any shape produces the final solution. This final solution corresponds to the feasible solutions on the right-hand sides of Figure 5, Figure 6, and Figure 7. When a sub-pool is handled by an optimisation tool, then usually many solutions are produced. But if all of those intermediary solutions are injected into the solution process of the entire pool, then it might be time consuming. Therefore, as a compromise, the injection of final solutions is proposed. Following this logic, the shape in Figure 4 will yield two feasible solutions for the injection; the shape in Figure 5 will yield three feasible solutions for the injection. The number of injected solutions from the shapes in Figure 11 equals the number of sub-pools in a shape minus one, because the feasible solutions of the entire pool are not injected – the entire pool stops only when the time limit is crossed. The shape with seven assets in the right-hand side of Figure 11 corresponds to the uninterrupted solution of the entire problem from scratch, i.e. no warm starts. It does not inject anything which is emphasised in the picture. When the time limit is crossed, all parallel calculations stop and from all the shapes the best upper and lower bounds are chosen.

3.1.7 Sub-pools with multiple starts

GUROBI enables to run multiple starts (to use multiple feasible solutions when solving optimisation problems) Figure 12 demonstrates a parallel architecture when three sub-pools of assets are solved in parallel. Then three feasible solutions are added as multiple starts into the solution procedure of the entire pool. The procedure in Figure 12 can be represented by means of the Tetris notation (the right-hand side of the picture). Note that a smaller sub-pool does not have to be a subset of any larger sub-pool. The shape in Figure 12 is a combination of GIs with shapes: ([1,2,4], [1,2,3,4,5,6,7]); ([1,4,6,7], [1,2,3,4,5,6,7]); and ([2,3,4,5,6], [1,2,3,4,5,6,7]). All the sub-pools are run in parallel. If the entire pool is not run until it is injected with the three warm starts, then it is a case of a cold warm start. However, if there is no lack of resources, then the

entire pool can be run in parallel to those three sub-pools. Then the solution procedure is interrupted to be fed with the warm starts. In GUROBI this is implemented via callback functions. After the injection of the warm starts is implemented the solution procedure for the entire pool is resumed without destroying Branch&Bound trees. This can be useful in cases when the processing of multiple starts can be implemented faster than separate processing of each warm starts.

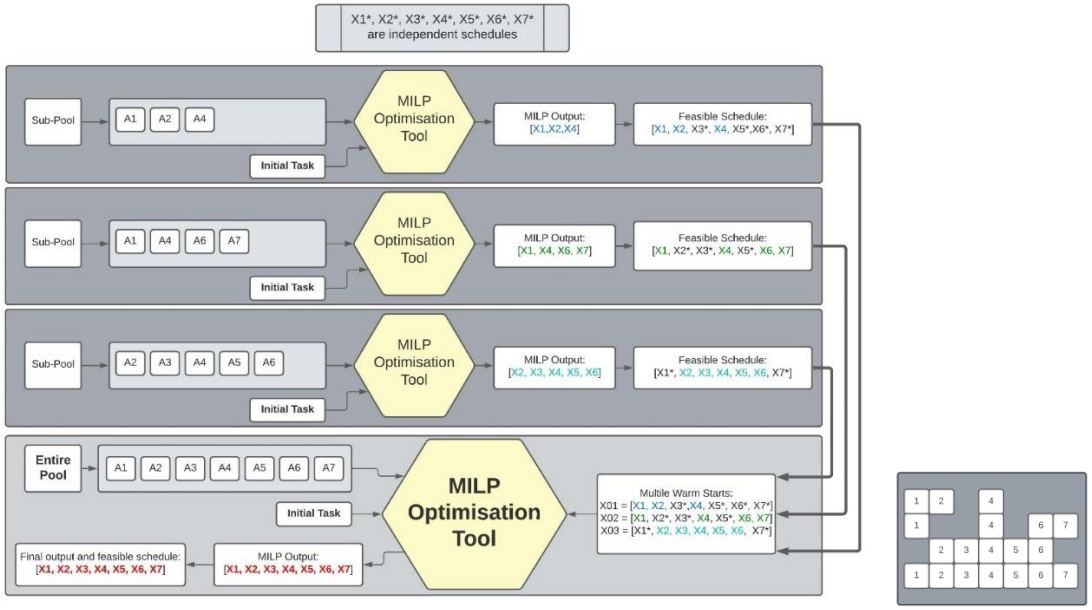

Figure 12 - GI with multiple starts.

3.1.8 Cold warmstart vs warm warmstart

Classical GI assumes cold warm starts: when a larger sub-pools takes the warm start generated by smaller sub-pools, the solution procedure starts from scratch (except for the warm start). This approach is reasonable when the number of parallel machines is limited. A single shape in GI can be managed by a single machine. However, it there are enough of computational resources, it is possible to perform a warm warmstart. If a shape contains 4 subsets, then it is possible to apply four parallel machines. Let us recall the shape from Figure 6: ([2,4], [1,2,4], [1,2,3,4,5], [1,2,3,4,5,6,7]). In this shape there are four sets. So, four machines start from scratch at the same time. When [2,4] finds a feasible solution, then [1,2,4] is paused and fed with the feasible solution [X1*, X2, X3*, X4, X5*, X6*, X7*] by a callback function. Then the solution procedure for [1,2,4] is resumed. The rest, i.e. [1,2,3,4,5] and [1,2,3,4,5,6,7] are not interrupted. When [1,2,4] finds a feasible solution, then [1,2,3,4,5] is interrupted and by a callback function is fed with the feasible solution [X1, X2, X3*, X4, X5*, X6*, X7*]. Then the solution procedure for [1,2,3,4,5] is resumed. When [1,2,3,4,5] finds a feasible solution, then [1,2,3,4,5,6,7] is interrupted to be fed with [X1, X2, X3, X4, X5, X6*, X7*]. The solution procedure for [1,2,3,4,5,6,7] is resumed until the time limit. Figure 13 visualises this shape in its middle-left part. For the shape in Figure 5, three parallel machines are required – three hexagons in Figure 13 (middle-left part). Figure 13 demonstrates parallel handling of 6 shapes. Those shapes require 17 parallel machines. In addition, seven machines are needed to calculate independent schedules. Then of those 5 shapes the best upper and lower bounds are chosen.

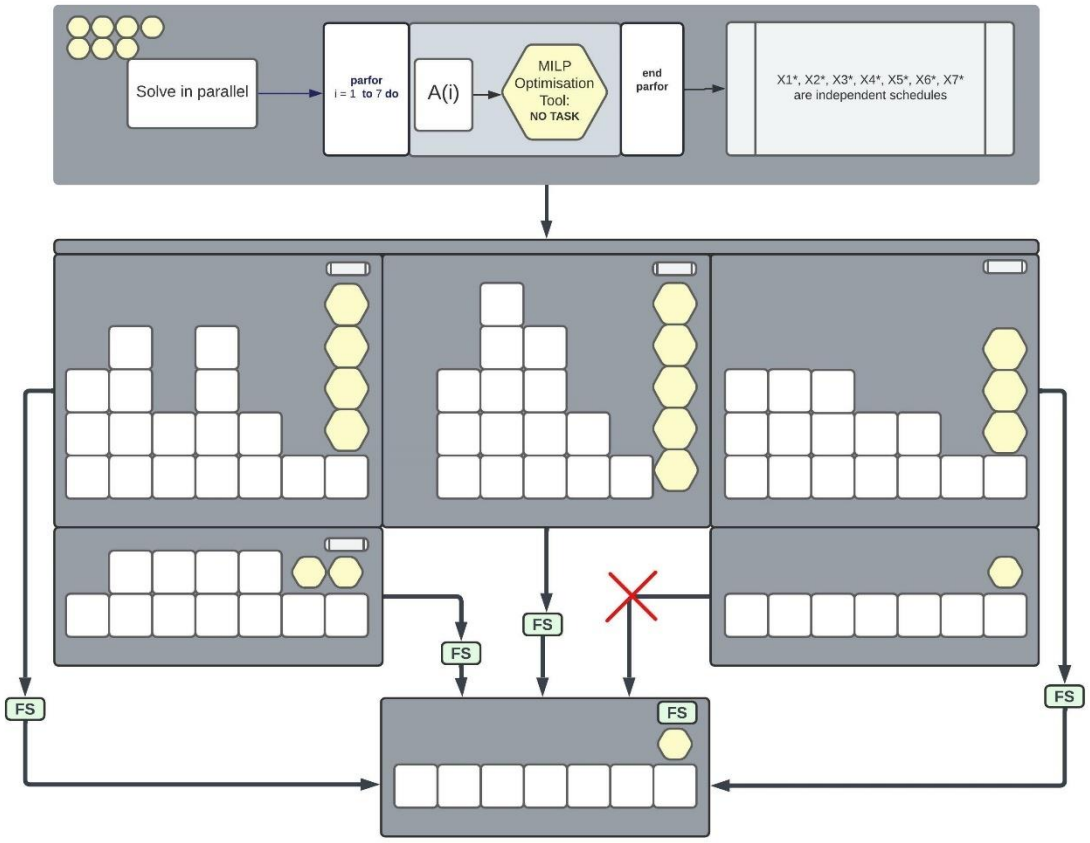

Figure 13 - Schematic visualisation of parallel handling of different *shapes*. Warm warmstarts.

3.1.9 Upper bound and MIPgap

No optimisation strategy can yield a better objective value of Problem (16)—(19) than $\sum_{k=1}^{\#A} p_k^T \cdot x_k^*$. Therefore, this value is an upper bound. If $UB_i, i \in \Xi$ are upper bounds obtained by solving different shapes indexed by the elements in set Ξ, then the final upper bound is obtained as follows:

$$\mathrm{ub}^* = \min\left(\sum_{k=1}^{\#A} p_k^T \cdot x_k^* \, , \min_{i\in\Xi} UB_i \right).$$

In a similar manner, the lower bound is obtained:

$$\mathrm{lb}^* = \max_{i\in\Xi} LB_i.$$

The final MIPgap is calculated as follows:

$$\mathrm{MIPgap} = \frac{\mathrm{ub}^* - \mathrm{lb}^*}{\mathrm{lb}^*} \times 100\%.$$

The closer the value of the MIPgap to zero, the more reliable the solution is. When MIPgap is zero then it is guaranteed that the incumbent is optimal. But in practical application this value should be a fraction of a per cent. For example, if the MIPgap is below 0.01% then the solution procedure is stopped (e.g., in GUROBI) and the optimal solution is claimed to be found. This, of course, relates to the default parameters. It is possible to set lower stopping criteria, e.g., stop if MIPgap is below one thousandth of a percent. However, in large-scale optimisation problems

there is no choice but to tolerate higher values than 0.01%. In the large-scale problems presented in this study, the MIPgap never crossed the value of 5%.

3.1.10 The Modified Gradual Increase

In classic GI, the capacity issue is a problem. For example, if an intermediary problem for a sub-pool $SubSet(i)$ has the following limitation:

$$\sum_{k \in SubSet(i)} A_k^T \cdot x_k \leq c,$$

and $a > c$, then the coupling constraint $\sum_{k \in SubSet(i)} A_k^T \cdot x_k \geq a$, cannot be satisfied with the assets from $SubSet(i)$. In this case $SubSet(i)$ will produce nothing and the time will be lost on checking that the intermediary problem for $SubSet(i)$ is infeasible. This holds for the initial formulation of GI. However, the introduction of a slack variable y solves this issue. The coupling constraint in every sub-problem is reformulated as follows:

$$\sum_{k \in SubSet(i)} A_k^T \cdot x_k + y \geq a,$$

and the utilisation of this variable is penalised in the objective function. Table 5 demonstrates the reformulation of the entire problem by the usage of the slack variable y. In the modified GI all such sub-problems are reformulated in a way demonstrated in the right-hand side of Table 5.

Table 5 - Modified GI – slack variable

Comparison of the Classic GI with the Modified GI	
Classic GI problem formulation	**Modified GI problem formulation**
$\text{maximise} \sum_{k=1}^{\#A} p_k^T \cdot x_k$ $\text{s.t} \sum_{k=1}^{\#A} A_k^T \cdot x_k \geq a,$ $B_k \cdot x_k \leq b_k$ $x_k(I) \in \{0,1\}, \forall\, k \in \{1,2,3,\dots,\#A\}$	$\text{maximise} \sum_{k=1}^{\#A} p_k^T \cdot x_k - \delta \cdot \lvert\lvert y \rvert\rvert_1$ $\text{s.t} \sum_{k=1}^{\#A} A_k^T \cdot x_k + y \geq a,$ $B_k \cdot x_k \leq b_k,\ y \geq 0,$ $x_k(I) \in \{0,1\}, \forall\, k \in \{1,2,3,\dots,\#A\}$

where $\lvert\lvert y \rvert\rvert_1 = \sum_{i=1}^{n} \lvert y_i \rvert$, $n = \dim(a)$, and δ is a positive number which is a penalty for the violation of the coupling constraint.

Figure 14 demonstrates the case when the task is large and is close to the entire capacity of the whole pool. The capacity of any sub-pool is not sufficient to perform the task. However, the modified GI is able to handle this problem and it produced a solution with MIPgap = 1.2% in 15 minutes while the entire pool with the classic formulation was not able to produce a feasible solution in this time limit.

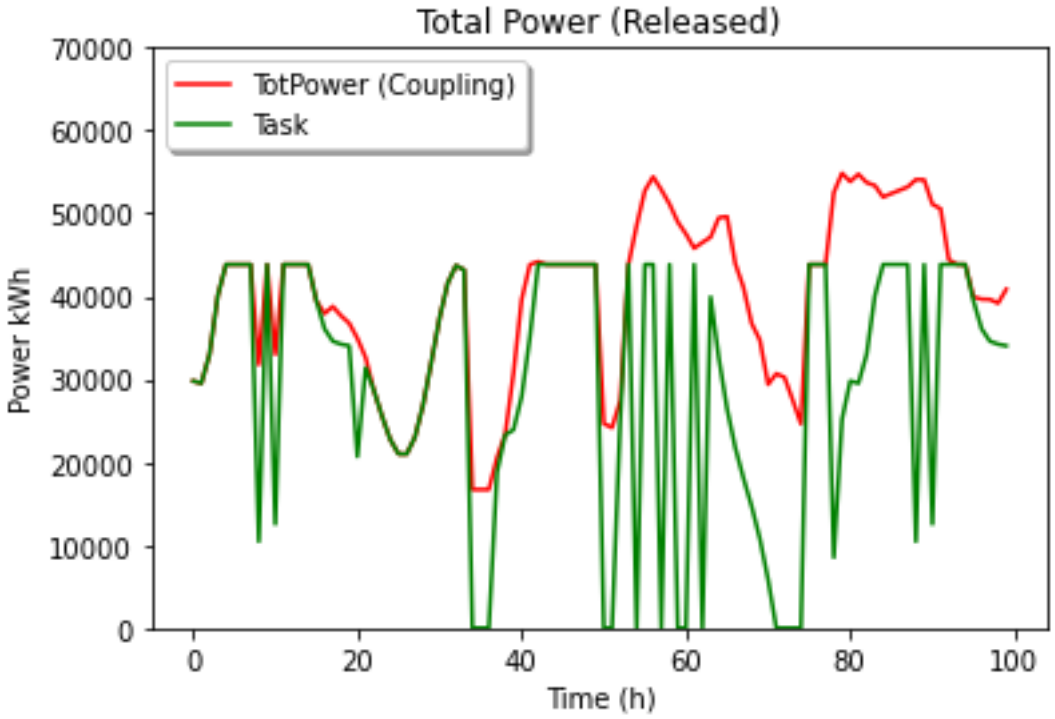

Figure 14 - The case where no sub-pool can solve entire problem, but all assets are able to handle the task. Classic GI can not help in this case, but it is doable by the modified GI.

3.1.11 Optimisation against prices.

Figure 15 visualises the total production against prices: the left part of Figure 15 demonstrates the cumulative production of the problem mentioned in Figure 14. In the right part of Figure 15, there is the visualisation of the cumulative productions of the independent schedules. When price is high, the production activates except for the cases when the balancing task is significantly large, as the right part of Figure 15 demonstrates.

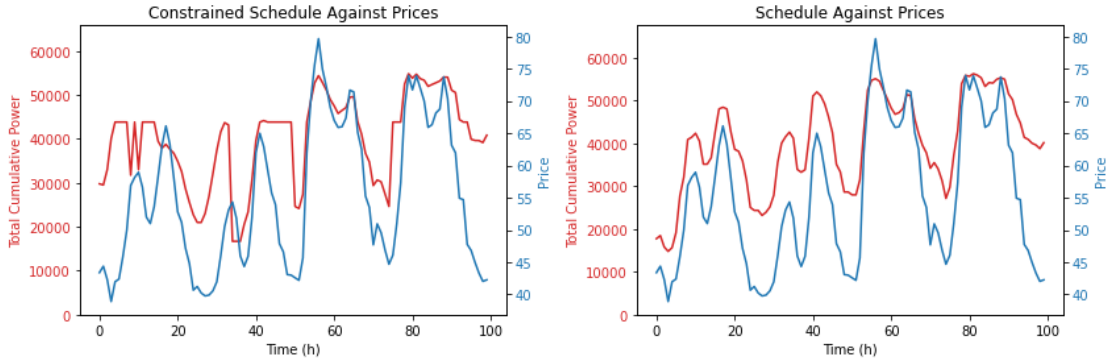

Figure 15 - The schedules against prices. The left-hand side part is the total cumulative power in the constrained optimisation problem visualised against prices. The right-hand side part is the cumulative sum of independent schedules visualised against prices.

3.1.12 Dependence on storage levels

Figure 16 and Figure 17 show schedules that depend on the initial storage levels: full versus empty. Figure 16 visualises the storage levels over 100 hours of the problem mentioned in Subsection 3.1.10: the blue line corresponds to the storage levels with the full initial level. The

red line is the diagram with the initial empty storage level. Note that the blue and red lines begin to overlap after a certain point.

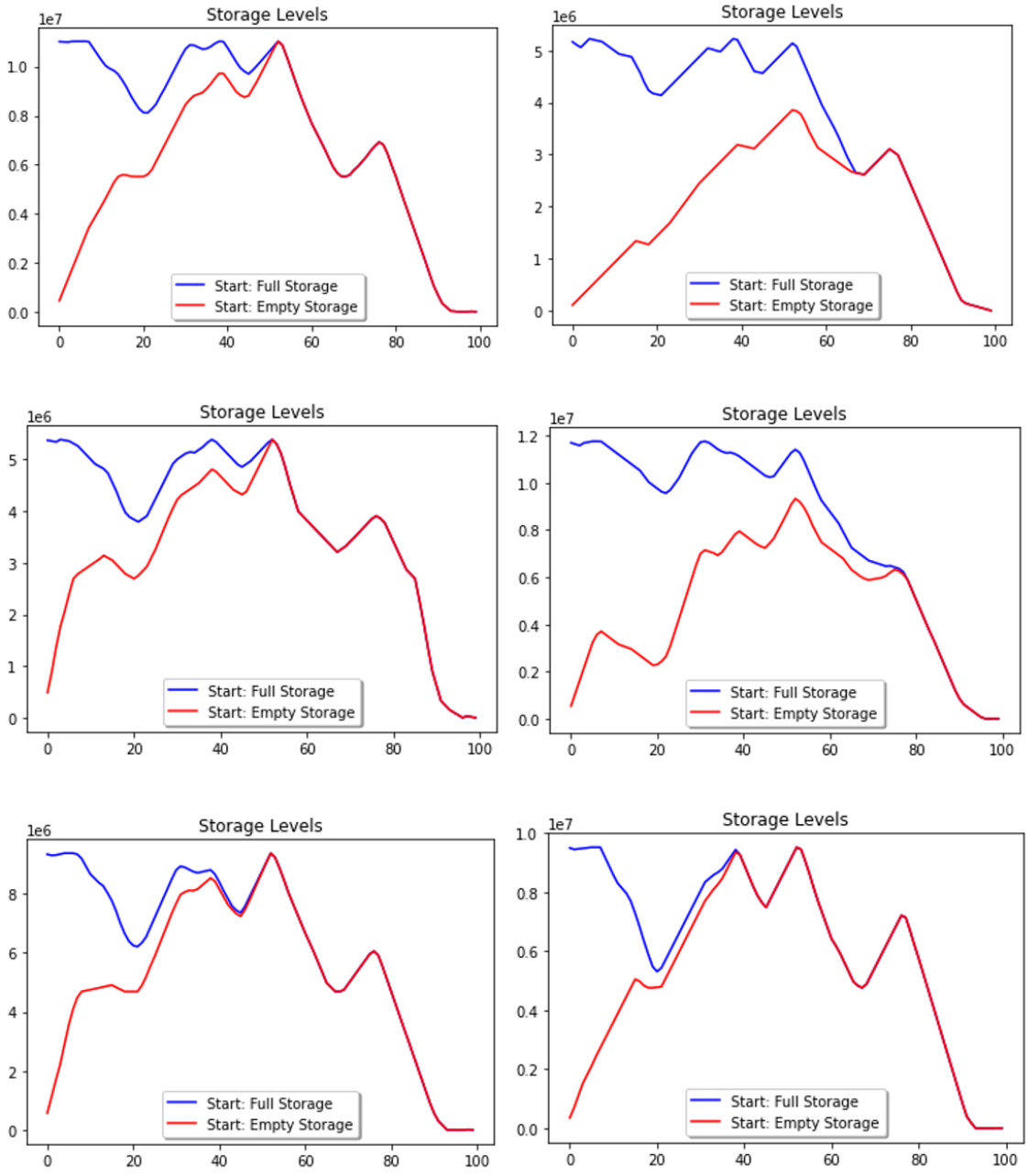

Figure 16 - The convergence of the optimal solutions (example with storage levels). This feature can be used for the acceleration of calculations: stage-wise decompositions. The start at the full and empty storage levels.

The overlap of the lines in the right part of Figure 16 occurs earliest (around hour 40), and in the right part it occurs latest. Figure 17 shows the overlap of a problem with a portfolio of biogas plants. The assets in Figure 17 are real, so the graphs are normalised. The assets in Figure 14 to Figure 16 have randomly chosen parameters i.e., are hypothetical, so they do not need to be normalised. Figure 17 shows the total productions at prices (upper part), the total schedules for the initially full and empty storage levels (the lower part on the left of Figure 17), and the storage levels. The upper part of Figure 17 shows that total production tends to take place when prices are high, and more fuel allows to benefit more from higher prices. The lower part of Figure 17 shows the overlap of the schedules after a certain point: total production and storage levels.

This overlap that occurs after a certain point can be used to generate first feasible solutions: the price vector can be used to estimate when the storage level will be low. It is reasonable that this happens after several periods of high prices. In Figure 17, this may be (upper part blue line) around hour 25-30.

The first feasible solution of the problem related to Figure 17 was found as follows: solve the problem $(T = 50)$ for the first 30 periods, i.e., the storage level will be zero at the end $(T = 30)$; take the values of $t = 25$ from the solution of the previous problem as the initial state for the problem starting from $t = 26$ to $t = 50$. Then merge the solution of these two problems into a feasible solution of the problem from $t = 1$ to $t = 50$. Each of the above problems has a smaller size than the problem with horizon T = 50. This approach allowed the corresponding problem to be solved 2.32 times faster, because the state at time $t = 25$ was close to the corresponding state it was the optimal solution of the original problem with $T = 50$.

Investigating how to exploit these overlaps could be the subject of a whole research project, and it is possible that methods based on dynamic programming will be useful in this regard.

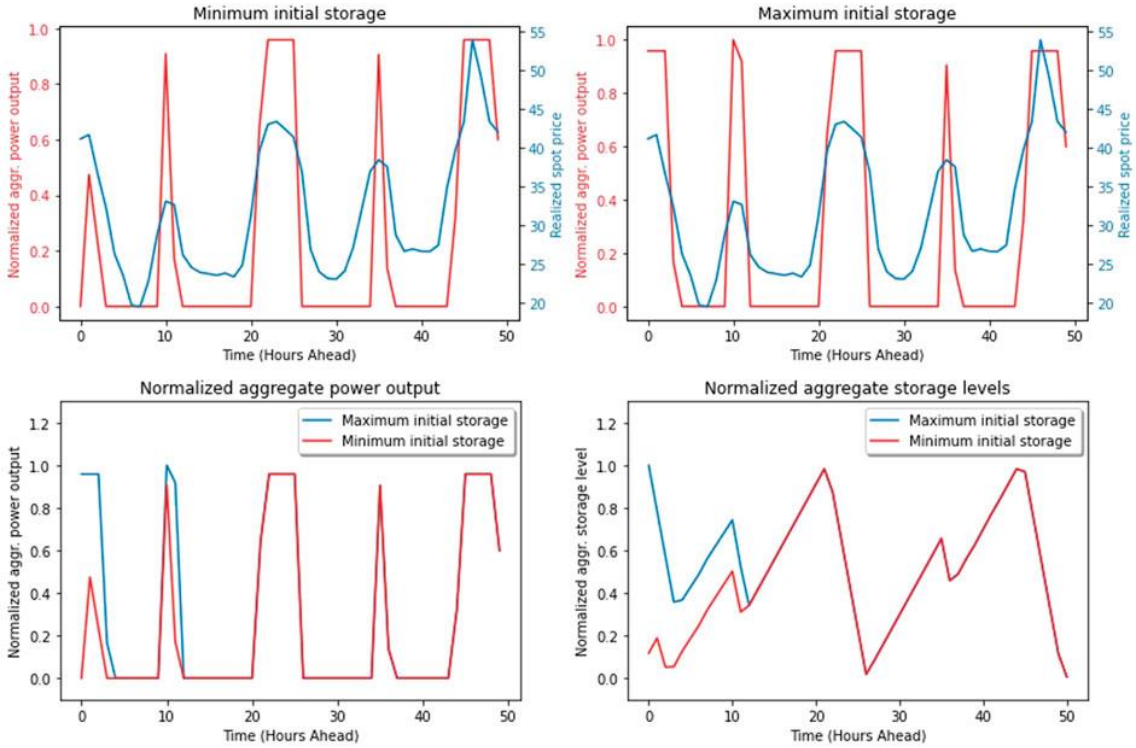

Figure 17 - The dependence of the total power output on forecasted prices and initial storage levels (Example from [27]).

4. Model Predictive Control

Model Predictive Control is a feedback control technique that naturally incorporates optimisation [3], [14], and [15]. In this study we consider certainty equivalent MPC and robust MPC proposed in [15]. In certainty equivalent MPC, we replace random quantities with predictions and solve the associated optimisation problem to produce the schedule over the selected planning horizon.

After optimisation, the first power schedule is executed i.e., the one associated with the time of optimisation. For the next step, we repeat this process incorporating the updated information about price and imbalance forecasts. Following the notation from Table 6, our MPC algorithm is defined as follows:

Take the initial state $State_0$ of the system (storage levels and states of the turbines) as the first input.

for $t = 1$ **to** N **do**:

- ***Forecast*.** Make price and imbalance forecasts that will be used as inputs in the optimisation:

$$\boldsymbol{Frc_t} = [Frc_t, Frc_{t+1}, \dots, Frc_{t+H-1}],$$

$$\Delta\, prod_t^W = [\Delta\, prod_t^W, \Delta\, prod_{t+1}^W, \dots, \Delta\, prod_{t+H-1}^W].$$

- ***Optimise.*** Solve the dynamic optimisation problem:

$$VPP(Frc_t^{DA}, \Delta\, prod_t^W, State_{t-1}), \tag{25}$$

where $\boldsymbol{Frc_t}$ determines the objective value and $prod_t^W$ determines the right-hand side of the power production constraints. Solving this optimisation problem yields the decision vector $\boldsymbol{y_t} = [y_t, y_{t+1}, \dots, y_{t+H-1}]$.

- ***Execute.*** We execute only y_t from the whole vector $\boldsymbol{y_t}$ because this decision relates to the most up-to-date time step and discard the rest of the components of $\boldsymbol{y_t}$.

- Determine the value $Revenue_t$ which is the revenue associated with the execution of y_t which equals:

$$Revenue_t = VPP_{rv}^h(Spt_t^{DA}, Spt_t^{ID}, \nabla\, prod_t^W)$$

and the next state is:

$$State_t = f(State_{t-1}, y_t, \nabla\, prod_t^W), \tag{25A}$$

where f is a linear function which determines the updates of the storage levels and states of the turbines/batteries according to (4), (6), (7), (9).

end for

Thus, the ultimate goal is the maximisation of the sum:

$$TotRev = \sum_{t=1}^{N} Revenue_t \rightarrow \max$$

i.e., the solution of the problems $VPP(\cdot,\cdot,\cdot)$ is an intermediate goal aimed at maximising the value $TotRev$. And in this study the model is assessed in terms of the value of $TotRev$ and the total speed-up. All the methods only differ by the approach to problems $VPP(\cdot,\cdot,\cdot)$ and the following will be applied:

- **Generic Approach**: the reliance on the power of the *solver* without any decomposition, splitting, parameter tuning, or a warm start.
- **Gradual Increase**: asset-wise decomposition is performed as is shown in the description of the method. In a similar way, it is possible to apply this approach on the turbines within each biogas power plants.
- **Proximal Jacobian ADMM**: the start from an LP relaxation to ensure that the coupling constraint is satisfied which is followed by the application of MILP methods to sub-pools (most on single power plants) to extract an integral solution.
- **Partial Integrality**: at time t, we relax all integral constraints of all variables $y_{t+1}, y_{t+2}, \dots, y_{t+H-1}$. This leads to a significant reduction of binary variables while providing a feasible action y_t . (Variable y_t is defined in Table 2.)
- **Parameter tuning with MPC**: within MPC, we can run parameter tuning of the *solver* after we have solved a problem. It can be conducted parallel to the solution of the new problems. After solving 20 problems, we change the tuning parameters which provides further speed-up. In Python's GUROBI environment, this can be conducted by means of the operation **model.tune( ).**
- **Hybrid of GI and Proximal Jacobian ADMM**: instead of adding single power plants in the pool they can be added by blocks. To solve the block subproblems faster, the method of *Gradual Increase* can be applied.

4.1 Robust Model predictive control (RMPC)

The difference between MPC and RMPC lies in a different approach to the second step of the algorithm (*Optimise*). In (29), there is a single forecast of prices and imbalances. In RMPC however [15], a predefined number of scenarios M is used, i.e.

$$Scenario_m = (Frc_t^{DA,m}, \Delta\, prod_t^{W,m} \quad \text{for } m = 1,2,\dots,M,$$

and use each scenario m to solve

$$\begin{aligned} &\text{maximise} \sum_{m=1}^{M} VPP(Frc_t^{DA,m}, \Delta\, prod_t^{W,m},\ State_{t-1}) \\ &\text{s. t.} \\ &p_{\tau,k,i}^{m^*} - p_{\tau,k,i}^{m^{**}} = 0 \quad \forall\, k,i \text{ and } \forall\, m^*, m^{**} \le M \text{ and } \forall\, \tau \le h. \end{aligned} \tag{2}$$

where the sum in (30) means that all the objectives are added up from the problems $VPP(Frc_t^{DA,m}, Frc_t^{ID,m}, \Delta\, prod_t^{W,m}, State_{t-1})$ for each m and maximise their sum; as for the constraints, they all are added into the set of the constraints of the resulting problem. The equality constraints in (30) ensure that the power produced within each scenario $Scenario_m$ will be the same for all the scenarios up to time h. This also ensures that the execution in the MPC phase i.e.,

$$Revenue_t = VPP_{rv}^h(Spt_t^{DA}, Spt_t^{ID}, \nabla\, prod_t^W)$$

will be the same for all the scenarios. The rest of the steps of the MPC algorithm remain unchanged. It can easily be shown that the same power values for all scenarios also imply the same states of the turbines and the same storage levels for all the scenarios when $t < h$. Nuances of Optimisation

The optimisation is against prices which implies that turbines should produce when the price is high and be off when it is low. As for the battery, it should discharge when the price is high, and it should charge the battery when it is either low or negative. Note that before the optimisation, the spot prices are not known and thus the decision-making relies on forecasts.

4.2 RMPC and the philosophy of Gradual Increase

Robust Model Predictive Control involves many scenarios. Their adding can be performed as in *Gradual Increase*. If a pool of assets is optimised over M scenarios, then it can first be optimised over a smaller number of scenarios m_0 $(m_0 \ll M)$ and then minor adjustments enables to produce a feasible warm start for the pool of M scenarios. Thus, the following problem is solved:

$$\text{maximise} \sum_{m=1}^{m_0} VPP(Frc_t^{DA,m}, \Delta\, prod_t^{W,m},\ State_{t-1})$$

$$\text{s.t.}$$

$$p_{\tau,k,i}^{m^*} - p_{\tau,k,i}^{m^{**}} = 0 \quad \forall\, k,i \text{ and } \forall\, m^*, m^{**} \leq m_0 \text{ and } \forall\, \tau \leq h,$$

and its output $y(\tau,k,i) = p_{\tau,k,i}^{1}\ \forall\, k,i,\tau \leq h$ is stored.

Then:

parfor $m = m_0 + 1$ to M **do**

Solve $VPP\left(Frc_t^{DA,m}, \Delta\, prod_t^{W,m},\ State_{t-1}\right)$ s.t. $p_{\tau,k,i}^{m} - y(\tau,k,i) = 0 \quad \forall\, k,i \text{ and } \forall\, \tau \leq h,$

end parfor

The concatenation of these solutions yields a feasible solution for the entire set of scenarios. This approach is visualised in Figure 18 for the following system of scenarios: $([1,2,3],[1,2,3,4,5],[1,2,3,4,5,6,7,8])$ i.e., there are 8 scenarios, but the start is with the first three scenarios, then the first five with a warm start, and then all the scenarios are employed with the warm start from the first five scenarios:

5. Data inputs into optimisation

As the objective function (1) and coupling constraints (15) suggest, the optimisation requires data inputs i.e., forecasts, and is conducted sequentially: at the start of the new period, Problem (25) is solved taking into account the state of the system. Then the real prices and imbalances become known, and the solution of Eq. (25A) enables us to calculate the real revenue yielded by solving Eq. (25) Then the state of the system is updated by (25A), and the process resumes.

Hence, the following data inputs are required:

- Historical prices on the EEX market and associated historical forecasts
- Historical wind production from the considered wind parks and associated historical forecasts

As for price inputs, we apply forecasts. Since these commercial forecasts are not available for a large audience, three forecasting methods based on *Deep Learning* are proposed whose implementation is possible using open-source software. The following subsection specifies what price forecasts are employed apart from the commercial ones

5.1 Price Inputs

Following developments in machine learning research for time series forecasting, we consider several state-of-the-art machine and Deep Learning approaches for the modeling of electricity prices:

- Multivariate Singular Spectrum Analysis (mSSA)—a popular and widely used time series forecasting method. As demonstrated in [1], mSSA was found to outperform deep neural network architectures such as LSTM and DeepAR in the presence of missing data and noise level.
- DeepAR—a methodology (developed by Amazon Research) for producing accurate probabilistic forecasts, based on training an auto-regressive recurrent network model [20]. We use GluonTS [2] implementation of DeepAR for our experiments.
- N−BEATS—a deep neural architecture based on backward and forward residual links and a very deep stack of fully connected layers [19]. The architecture demonstrated good performance in M4 forecasting competitions and more recently [19] has been used for electricity load forecasting.

5.2 Production Inputs

Historical imbalances are obtained as the difference between the real aggregate production and the forecasted aggregate production. The aggregate production for both cases is obtained by adding up the productions from each wind turbine. For the aggregate production of wind, commercial forecasts are used. The prediction of imbalances would exceed the scope of this deliverable and in this study, the imbalances are treated as follows:

- Usage of historical imbalances as inputs—an assumption of a perfect foresight.
- Fitting imbalances with an ARMA process and the simulation of them within an RMPC scheme

In the exploration of the speed of the optimisation algorithms, the commercial price forecasts are employed, and the imbalances are assumed to be known in advance. In the exploration of the precision, all proposed price forecasts are employed and where imbalances are assumed to be uncertain, the optimisation is conducted within RMPC.

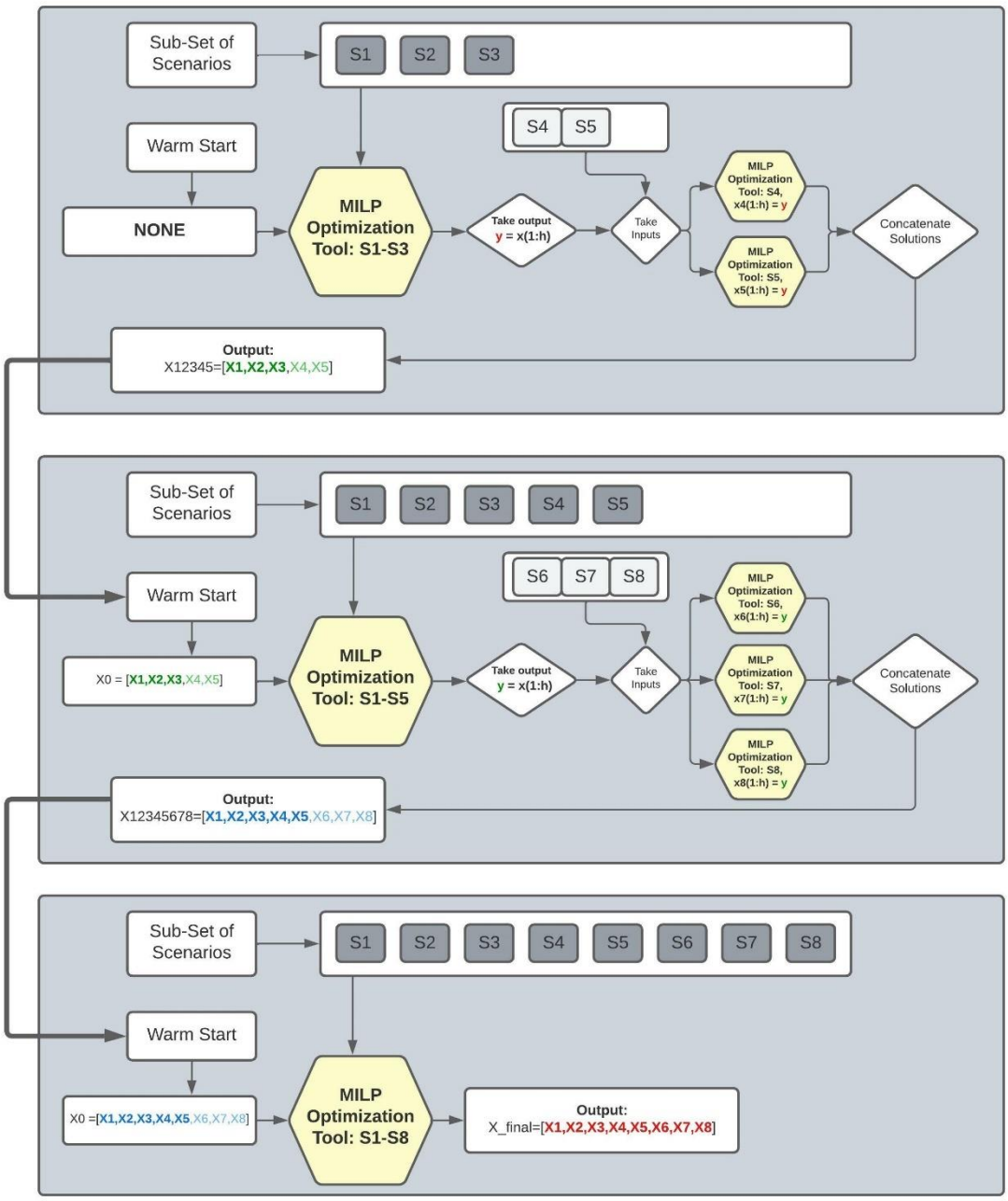

Figure 18 - RMPC implemented as in Gradual Increase: instead of gradual addition of assets to a pool, scenarios are gradually added to a set of scenarios. It involves minor adjustments.

6. Nuances of the optimisaton

The optimisation is against prices which implies that turbines should produce when the price is high and be off when it is low. As for the battery, it should discharge when the price is high, and it should charge the battery when it is either low or negative. Note that before solving the optimisation problem, the spot prices are not known and thus the decision-making relies on forecasts.

6.1 The Choice of assets for the initial Sub-Pool

In [27] it is suggested choosing such assets according to their contribution to the coupling constraint from a previously solved problem (when problems are solved in real-time). However, this is an approach based on simple rules. A more detailed analysis of such choices is desirable: often this choice is tailored to specific features of the assets. As Table 7 (columns GI and GI_mod) suggests, the choice of sub-pools is important because the number of computational tools is often limited and not always there is enough of resources to implement GI with warm warmstarts.

6.2 Hardware and solvers

We use m5d.4xlarge EC2 instance within Amazon Web Services i.e., 16 vCPUs and 64 GiB RAM. In optimisation, two *solvers* are employed: CBC and GUROBI. To optimise our set of assets, we can get by with CBC, but in order to deal with portfolios of replicated biogas power plants, it is necessary to resort to the commercial *solver* GUROBI.

6.3 Implementation of Proximal Jacobian ADMM

The proximal Jacobian ADMM (Alternating Direction Method of Multipliers) was developed in [8] and [23] and its convergence for linear programming problems was proven in the same literature. In this study apart from *Gradual Increase*, we also use Proximal Jacobian ADMM, where we first relax all integrality constraints and let the algorithm iterate until the acceptable violation of coupling constraints is achieved: the preservation of the rest of the constraints is provided by the subproblems. After finishing the linear programming part, we get the action vectors x_k^{LP}, for each $k \in \{1,2,\dots,\#A\}$. Let a_k^{LP} denote $A_k \cdot x_k^{LP}$ i.e., $a_k^{LP} = A_k \cdot x_k^{LP}$. Then for each asset k, we recover the mixed-integer action vector by solving the following subproblems.

$$\begin{aligned} &\text{maximise} \quad p_k^T \cdot x_k \\ &s.t. \;\; A_k \cdot x_k = a_k^{LP}, \\ &\qquad B_k \cdot x_k \le b_k, \\ &\qquad x_k(I) \in \{0,1\}. \end{aligned} \tag{26}$$

Sometimes these problems are infeasible, but we handle this as follows: if K is a subset of $\{1,2,\dots,\#A\}$ such that for each $k \in K$, problem (31) is infeasible, then we can solve:

$$\begin{aligned} &\text{maximise} \quad \sum_{k\in K} p_k^T \cdot x_k \\ &s.t. \;\; \sum_{k\in K} A_k \cdot x_k \ge \sum_{k\in K} a_k^{LP}, \\ &\qquad B_k \cdot x_k \le b_k, \\ &\qquad x_k(I) \in \{0,1\}, \\ &\qquad \forall k \in K. \end{aligned}$$

In all our problems $\#K \leq 4$ and such pools are solved in matter of seconds. However, this procedure provides sub-optimal solutions, but this technique is amenable to parallelisation [23] and can handle extra-large pools of assets. It can also be easily shown that this algorithm can be combined with *Gradual Increase*. When the number of biogas power plants exceeds 300, memory constraints affect the solution. However, in the case of ADMM the dependence of memory on the number of assets is linear therefore the linearisation of the problem and the solution of it by proximal Jacobian ADMM can be used for generating sub-optimal mixed-integer solutions. Thus, it is possible to create bigger clusters of assets with their own coupling constraints

$$\sum_{k \in Cluster(j)} A_k \cdot x_k \geq \sum_{k \in Cluster(j)} a_k^{LP}, \qquad \text{for } j \in \{1,2,3,\dots,\#\text{Clusters}\},$$

and to apply GI to every cluster. Figure 19 demonstrates this clustering.

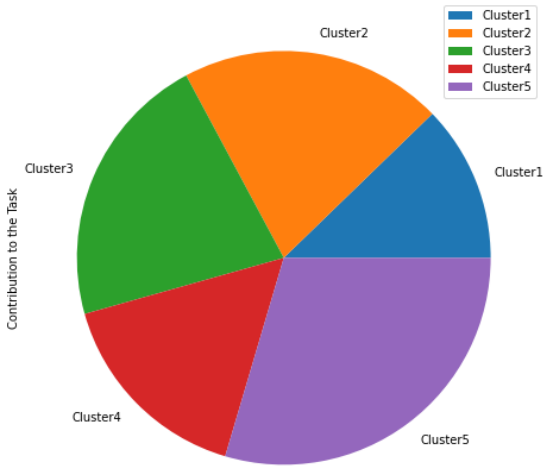

Figure 19 - A feasible solution obtained by Proximal Jacobian ADMM enables to determine the contribution of each asset to the coupling constraint. It is then possible to take clusters of assets using those contributions and then to handle them by mixed integer linear programming.

Table 6 - Notation for Model Predictive Control

The Symbol	Explanation
H	the prediction horizon
h	the execution horizon (realised schedule)
$\nabla prod_t^W$	the realised imbalance
$\boldsymbol{\nabla prod_t^W}$	the forecasted imbalance for time t $\boldsymbol{\nabla\, prod_t^W} = [\Delta\, prod_t^W, \Delta\, prod_{t+1}^W, \dots, \Delta\, prod_{t+H-1}^W]$
$\Delta\, prod_t^W$	the forecasted imbalance for time t
$\boldsymbol{\Delta\, prod_t^W}$	the forecasted imbalance for times from t to $t + H - 1$
$\boldsymbol{State_t}$	the state of the system at time t: storage levels, and the states of turbines.

7. Results

This section presents the results of the calculations related to the speed (Optimality 7.4), forecast precision in terms of MSE (Subsection 7.1), and the optimality in a stochastic environment (Subsection 7.3). In both cases, we consider the following period: 2020/1/1–2020/9/26. In the case of the speed, the prediction horizon (H) is 2 days ahead. In the case of optimality, our prediction horizon (H) is 3 days ahead. In both cases, the execution horizon (h) is 24 h. Price forecasts are updated once or two times per day, but production forecasts are updated every 15 min. Therefore, it is crucial to be able to solve every optimisation problem in maximum 15 min, otherwise it will not be possible to deploy these algorithms for real-time applications. We also used commercial price forecasts and a price forward curve whose origin will not be disclosed due to confidentiality issues.

In the case of the speed, only state-of-the-art commercial price forecasts are used, and perfect knowledge of the imbalance is assumed. We emphasise that the size of imbalance increases proportionally to the number of assets; otherwise, the optimisation problem would be trivial. In addition, in the case of the speed, timing constraints are introduced and for each turbine, UT and DT are generated by means of a uniformly distributed random variable taking values: 0, 1, 2, 3, 4. In the case of optimality, we consider different forecast methods described in Section 5 affect the cumulative revenue over the period.

7.1 Analysis of Price Forecasts

We apply the above models to generate and benchmark forecasts 168-h ahead at three time points: 1) 31-Dec-2020 2) 31-Jan-2021 3) 29-Feb 2021. Our approach is to demonstrate benefits from using machine and Deep Learning forecasting technologies, rather than a comprehensive evaluation of different classes of methods or to add to the debate on the benefits of machine learning vs. statistical algorithms. We generate forecasts for 168 h (7 days) ahead and benchmark forecasts against 168 h lagged naive forecasts that is able to capture daily and hourly dynamics of electricity prices (thus is a competitive benchmark in the short-term, especially as our forecasts are of parsimonious nature and only use historic price information).

We have utilised mean squared error (MSE) to measure the performance of considered forecasting frameworks. Our results demonstrate significant benefits (as measured by forecasting value added—FVA) from applying powerful Deep Learning frameworks such as DeepAR and N-Beats, as well as data-driven models such as mSSA that can capture dynamic and feature-rich behavior of electricity prices. The applied frameworks can demonstrate, even without any hyper-parameter optimisation, that powerful open-source frameworks such as mSSA, DeepAR and N-Beats are able to generate (see Table 8) very competitive forecasting inputs in comparison with expensive commercial forecast feeds.

7.2 Analysis of Results in Terms of Decision Variables

This subsection demonstrates results in terms of decision variables and input parameters. In the optimisation models presented in this deliverable, there are the following variables: p, u, v, w, soc (See Table 2). Instead of visualising all the variables, we concentrate on the most important ones. Let us note first, that the variables p and u (i.e. $p_{t,k,i}$ and $u_{t,k,i}$ $\forall t, k, i,$) are interrelated: for turbines, Eq. (3) implies that if $p_{t,k,i} > 0$ then $u_{t,k,i} = 1$, and if $p_{t,k,i} = 0$, then $u_{t,k,i} = 0$; for batteries these insights are similar.

7.2.1 The dependence of the revenue on the quality of the forecasts

The top left part of Figure 20 visualises a power schedule against corresponding spot prices under perfect foresight. The top right part of Figure 20 visualises a schedule against corresponding spot prices under the usage of commercial price forecasts. The forecasts of the spot prices were generated on 05. 30. 2020 at 12:00 AM and the price vector is 50 consecutive hourly price forecasts starting at 9:00 PM on 05. 30. 2020. Let us note that between hours 30 and 40 where there is a local maximum spot price, the schedule was adjusted to this maximum i.e., there is a spike in the aggregate power production. However, the commercial price forecasts have not

predicted the local maximum price at that point, therefore there is no spike in the power production in the left part of the picture. Let us also note that the spot price is negative in the following chunks:

- During hours 17 — 21.
- During hours 41 — 43.

In the top left and middle parts of Figure 20, the aggregate power output is negative at the chunks where the price is negative. This is enabled by Eq. 13, and when the price is negative then the battery is charged. The commercial forecasts have not predicted these negative prices. Instead, these forecasts values are close to zero which results in "doing nothing" at these chunks. This emphasises the importance of the quality of the forecasts.

The middle part of Figure 20 visualises the normalised aggregate schedules for the system where all the turbines have the following parameters of the timing constraints: $\mathrm{UT} = \mathrm{DT} = 12$.

The bottom part of Figure 20 visualises the normalised schedule of the first turbine of the fourth asset in the pool for perfect foresight. In the bottom-left part of Figure 20, there are no timing constraints. In the bottom right part of Figure 20, there is $\mathrm{UT} = \mathrm{DT} = 12$.

Let us note that any switch-on of a turbine is associated with a guaranteed loss of fuel. For example, let I denote a subset of all turbines of Asset k. Then the switch-on of all the turbines within I, yields the following guaranteed minimum usage of the fuel within the storage of Asset k:

$$\sum_{i \in \Xi} Pmin_{k,i} \cdot UT^{(k,i)},$$

which implies that the loss of fuel is proportional to UT. This is the reason why in the bottom right part of Figure 20, it was impossible to cherry-pick the local maximum spot price at time 36: the turbine was switched on at hour 21 and was on until hour 32. At hour 32, the storage was almost empty, therefore the turbine had to be switched off. In 3 hours, there was that local maximum price, but there was not enough fuel in the storage. In addition to this, $\mathrm{DT} = 12$ which makes it impossible to switch on that turbine. We have also calculated the schedule for $\mathrm{UT} = 12$ and $\mathrm{DT} = 0$. In this case, it is possible to switch on the turbine after hour 32, but it was not implemented, because this switch-on would enable us to cherry-pick the local maximum at time 36, which is 10.96 euros, but at hour 42, the price is −48.17 euros, and this "bonus" is inevitable if the turbine is switched on after hour 32.

In the top left part of Figure 20, there are no timing constraints and there is perfect foresight, therefore there is the largest revenue. In the top right part of this figure, there are no timing constraints, but the prices are not known. Hence the revenue there is 91.73% of the maximum. In the middle-left part of this picture, there is perfect foresight, but there are timing constraints i.e., $\mathrm{UT} = \mathrm{DT} = 12$. And the revenue for this case is 91.19% of the maximum. In the middle right part of Figure 20, there are timing constraints, and the prices are forecasted. The revenue for this case is 85.22% of the maximum.

Hence, in order to increase the revenues, apart from optimisation methods, it is important to:

- Improve price forecasts
- Perform technological advances on turbines to reduce UT and DT, if it is possible.

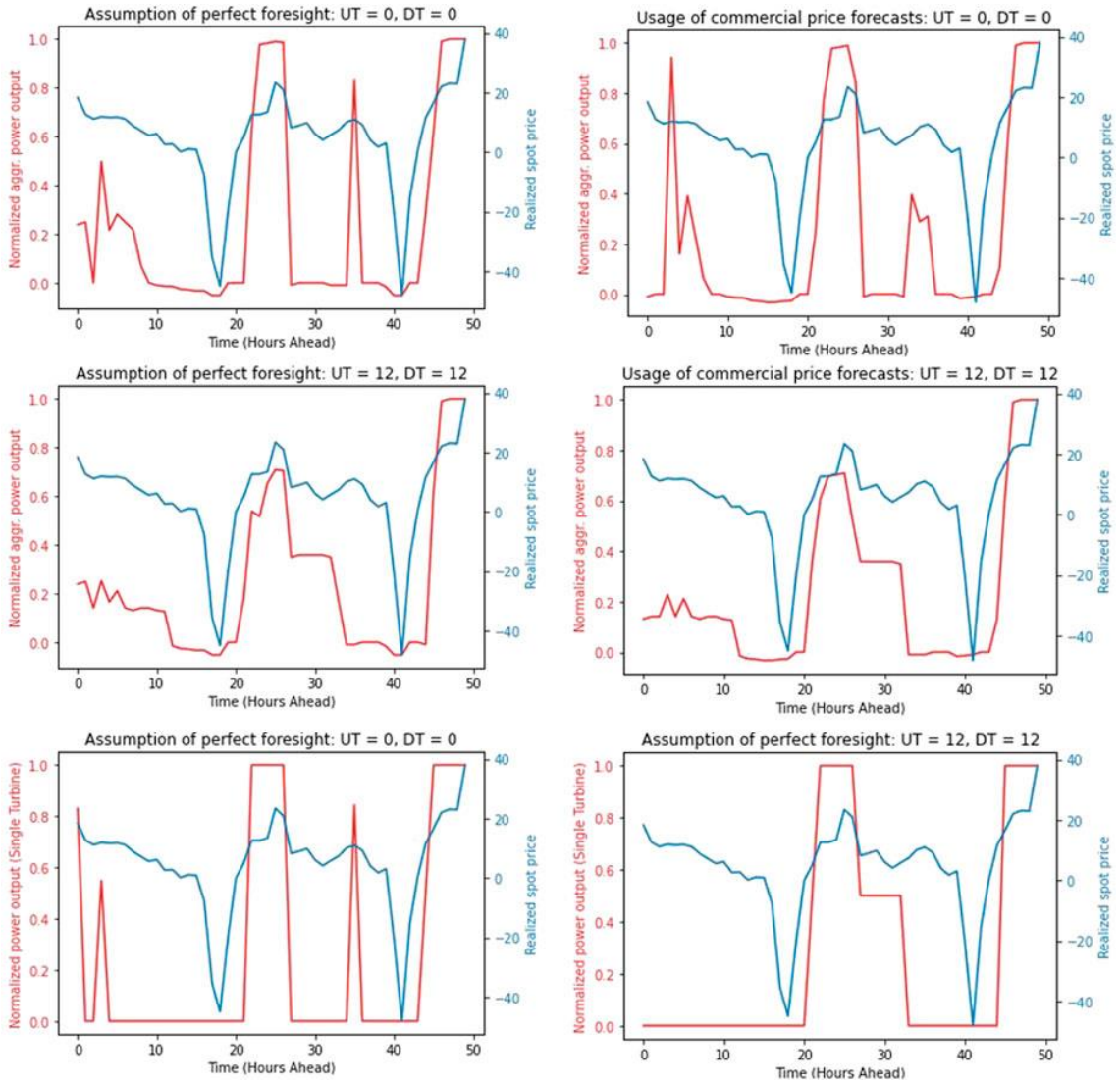

Figure 20 - The dependence of the revenue on the quality of the forecasts and timing constraints.

7.3 Speed

When we explore the speed, our benchmark is the time needed to solve the MILP problem by the GUROBI *solver*. We call this approach the *Brute Force* and the proposed decomposition and splitting algorithms must enable us to find the solution faster. Table 7 summarises the average calculation times provided by different methods. The headers of each column refer to the method applied and in the caption of the table there is the explanation of the abbreviations in the headers. The methods in the headers with italic font are the techniques which yield sub-optimal solutions but the resulting value of $TotRev$ yielded around 99.96% of the maximum. These methods are: partial integrality, partial integrality coupled with *Gradual Increase*, proximal Jacobian ADMM, and proximal Jacobian ADMM coupled with *Gradual Increase*. The methods such as Brute Force and *Gradual Increase* lead to the optimal solution, provided the *solver* has enough time. In Table 7, we can observe how *Gradual Increase* (GI) outperforms Brute Force (BF) in terms of time. When we solve problems by Brute Force, we only have the time limit of 15 minutes i.e., the computations finish if either the optimal solution is found (the relative difference between the upper and the lower bound is below 0.01%) or the time limit has expired. In columns BF, GI, *PI,* and *GI-PI*, we try to get the optimality message within 15 minutes. As for methods based on ADMM, this approach is irrelevant and the stopping criteria for these algorithms are described in Subsection 6.3. So, in Table 7, we can see that GI improves over BF when we increase the number of assets, but when we get closer to 200 then the difference is smaller. This is because we finish either when the optimal solution is found or when the time limit is exceeded. Thus, the more assets we

have in the portfolio, the more cases we face when the computation time would expire. As stated above, GI helps us find the optimal solution, but it has nothing to do with the confirmation of the optimal solution (calculation of upper bounds). To accelerate this confirmation, we can resort to *Partial Integrality*. In this case, we do not get to the expiration of the time limit and the fastest method turned out to be the combination of *Gradual Increase* and *Partial Integrality* which yields average calculation time 4.17 times smaller than that of *Brute Force*. The method *Proximal Jacobian ADMM* is relevant only for large pools of assets, therefore we started calculations from 52 assets in the pool. And we can see that *Brute Force* outperforms this method in terms of speed for 52, 102, and 150. But when we have 200 assets, then *Proximal Jacobian ADMM* outperforms *Brute Force*. When we combine *Proximal Jacobian ADMM* with *Gradual Increase*, then we get further acceleration (column ***GI-PJ-ADMM***) for 200 assets – approximately 1.3 times faster than *Brute Force*. Another positive side of *Proximal Jacobian ADMM* is that it imposes lighter memory requirements on the machine than other proposed methods [5]. In our further research, we are going to combine all these methods and to conduct pruning non-tight constraints by means of the methods of machine learning.

Table 7 is present in [27] without column GI-mod. This column was obtained by re-runs of the problems used in Table 7, solving the optimisation problems as follows:

- Modify the optimisation problems as is shown in Table 5.
- Use parallel computational units to solve four different shapes; one entire pool; one entire pool which is interrupted when each of sub-pools produces its final feasible solution as is shown in Figure 11.

Column GI_mod demonstrates average computation times which is on average by 1.26 times shorter.

7.4 Optimality

When we explore the optimality, we attempt to get as close as possible to the total revenue that we would achieve in the case of perfect foresight, when both prices and imbalances are known in advance. So, the cumulative revenue under perfect foresight is our benchmark and we explore the percentage of it provided by different methods. We explore this by means of variable σ (Defined in Subsection 2.7.5), which is apparently unit for the case of perfect foresight. Table 8 summarises the cumulative revenues. All these revenues are divided by total revenue yielded by the perfect forecast i.e., the variable $TotRev$ associated with perfect foresight defined in Subsection 4.1 i.e., Table 8 summarises the σ-values. It can be observed that when commercial state of the art forecasts are used and perfect foresight of productions is assumed, then RMPC yields slightly higher revenues. In the case of uncertain productions, only RMPC is utilised because we prefer simulations of the realisations of productions rather than trying to find a point forecast: the forecast of the imbalance is a challenging task and the exploration of it exceeds the scope of this report. Mimicking this imbalance by ARMA processes yields σ equal to 84.19% if we use commercial forecasts, and σ equal to 81.29%, if we use mSSA, which can be achieved by utilising open-source software. Table 8 is also demonstrated in [27] without the data for price forward curves. The usage of these forecasts yields slightly better values of TotRev than mSSA, N-BEATS, and DeepAR, but they are slightly worse than those of the commercial price forecasts and Table 8 shows it.

Table 7 - Average speed of calculations (in seconds) for methods: Brute Force (BF), Gradual Increase (GI), Modified Gradual Increase with 3 shapes (GI_mod), Partial Integrality (PI), Combination Gradual Increase&Partial Integrality (GI-PI), Prox-Jacobian ADMM (PJ-ADMM), Combination of Gradual Increase and PJ-ADMM (GI-PJ-ADMM).

#Assets	BF	GI	GI_mod	*PI*	*GI-PI*	*PJ-ADMM*	*GI-PJ-ADMM*
4	6	8	5	-	-	-	-
8	45	50	42	-	-	-	-
10	52	49	46	6	5	-	-
25	140	119	107	9	8	-	-
32	166	129	120	55	44	-	-
52	278	184	165	105	79	390	368
102	407	257	176	113	93	450	411
150	480	358	272	231	143	497	479
200	643	601	428	280	154	532	495

Table 8 - The σ-value for MPC/RMPC and different forecast methods (σ is the percentage of maximum possible revenue).

Price Input	Imbalance Input	Control Method	σ (in %)
Perfect Foresight	Perfect Foresight	MPC	100.00
Commercial	Perfect Foresight	RMPC	94.40
Commercial	Perfect Foresight	MPC	93.98
Price Foreward Curve	Perfect Foresight	MPC	90.23
mSSA	Perfect Foresight	MPC	88.75
Commercial	ARMA simulations	RMPC	84.19
Price Foreward Curve	ARMA simulations	RMPC	82.64
mSSA	ARMA simulations	RMPC	81.29
DeepAR	ARMA simulations	RMPC	77.34
N-BEATS	ARMA simulations	RMPC	72.53

8. Conclusion

In this report we have presented optimisation and forecast algorithms with the goal of more efficient management of the VPPs and the by-product of such algorithms is the facilitation of the integration of DERs into the grid. In other words, one of the ways to profit from owning a DER resource is connecting it to an efficiently optimised virtual power plant. The analysis in terms of decision variables revealed how timing constraints and the precision of price forecasts affect the revenue. We have explored the computation times of GI and Proximal Jacobian ADMM depending on the number of biogas power plants and demonstrated how parallelisation and further improvements in GI improve the speed of calculations. The main point of these calculations is that the presented algorithms can handle up to hundreds of power plants within a pool if we properly use decomposition and splitting methods (or their combination). We also learned that the combination of *Gradual Increase* with Partial Integrality yields further improvements in computation time. Similar results are achieved when we combine *Gradual Increase* with Proximal Jacobian ADMM. We have also conducted experiments with a pool of two biogas power plants and a single battery to compare different forecasts (Commercial state-of-the-art, mSSA, DeepAR, N-BEATS) and optimisation methods (MPC and Robust MPC). The usage of Robust MPC yielded the revenue 0.447% higher than that of MPC. Therefore, in our future research, we will consider combining Robust MPC with decomposition methods to achieve analogous results for larger pools. When the VPP is optimised by means of RMPC under price and imbalance uncertainty, then the utilisation of commercial price forecasts yields 84.19% of the maximum possible revenue whilst the usage of our mSSA forecasts yields 81.29% of it. However, this forecast can be achieved by using of open-source software. All the optimisation calculations related to Table 4 were conducted with CBC *solver*. The ability to run virtual power plants by means of open-source software also facilitates balancing and the incorporation of newly DERs into the grid. However, there is still a need for commercial *solvers* when we deal with large systems consisting of hundreds of assets.

9. List of Tables

10. List of Figures

11. References

[1] [Dataset] Agarwal, A., Alomar, A., and Shah, D. (2020). On multichannel singular spectrum analysis

[2] Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., et al. (2020). Gluonts: Probabilistic and neural time series modeling in python. Journal of Machine Learning, Research 21, 1–6

[3] Bemporad, A. (2006). Model predictive control design: New trends and tools. In Proceedings of the 45th, IEEE Conference on Decision and Control (IEEE)

[4] Bertsimas, D. and Stellato, B. (2020). The voice of optimisation doi:arXiv:1812.09991

[5] Boyd, S., Parikh, N., Chu, E., and Peleato, B. (2011). Distributed optimisation and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning 3, 1–122. doi:10.1561/22000000165

[6] Chen, G. and Li, J. (2018). A fully distributed ADMM-based dispatch approach for virtual power plant problems. Applied Mathematical Modelling 58, 300–312. doi:https://doi.org/10.1016/j.apm.2017.06.010.

[7] DCbrain (2020). Integrating green gases. how to reach new levels of performance with hybrid ai

[8] Deng, W., Lai, M.-J., Peng, Z., and Yin, W. (2017). Parallel multi-block ADMM with o(1 / k) convergence. Journal of Scientific Computing 71, 712–736

[9] Graabak, I. and Korpas, M. (2016). Balancing of variable wind and solar production in continental europe with nordic hydropower – a review of simulation studies. Energy Procedia 87, 91–99. doi: https://doi.org/10.1016/j.egypro.2015.12.362. 5th International Workshop on Hydro Scheduling in Competitive Electricity Markets

[10] Hewamalage, H., Bergmeir, C., and Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting 37, 388–427

[11] Hong, T., Pinson, P., Fan, S., Zareipour, H., Troccoli, A., and Hyndman, R. J. (2016). Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. International Journal of Forecasting 32, 896–913. doi:10.1016/j.ijforecast.2016.02.001

[12] Kaminsky, V. (2013). Energy Markets (Risk books)

[13] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100, 000 time series and 61 forecasting methods. International Journal of Forecasting 36, 54–74. doi:10.1016/j.ijforecast. 2019.04.014

[14] Mattingley, J., Wang, Y., and Boyd, S. (2011). Receding horizon control. IEEE Control Systems Magazine 31, 52–65.

[15] Moehle, N., Busseti, E., Boyd, S., and Wytock, M. (2019). Dynamic energy management. Springer, Large Scale Optimisation in Supply Chains and Smart Manufacturing , 69–126

[16] Morales-Espana, G., Gentile, G., and Ramos, A. (2015). Tight MIP formulations of the power-based unit commitment problem. OR Spectrum 37, 929–950. doi:https://doi.org/10.1007/s00291-015-0400-4

[17] Munsing, E., Mather, J., and Moura, S. (2017). Distributed optimisation and statistical learning via the alternating direction method of multipliers. IEEE Conference on Control Technology and Applications (CCTA) , 2164–2171doi:10.1109/CCTA.2017.8062773

[18] Nowotarski, J. and Weron, R. (2018). Recent advances in electricity price forecasting: A review of probabilistic forecasting. Renewable and Sustainable Energy Reviews 81, 1548–1568. doi:10.1016/j.rser.2017.05.234

[19] Oreshkin, B. N., Dudek, G., and Pelka, P. (2020). N-beats neural network for mid-term electricity load forecasting. ArXiv abs/2009.11961

[20] [Dataset] Salinas, D., Flunkert, V., and Gasthaus, J. (2019). DeepAR: Probabilistic forecasting with autoregressive recurrent networks

[21] Spiliotis, E., Makridakis, S., Semenoglou, A.-A., and Assimakopoulos, V. (2020). Comparison of statistical and machine learning methods for daily SKU demand forecasting. Operational Research , 1–25

[22] Tan, Z., Li, H., Ju, L., and Tan, Q. (2018). Joint scheduling optimisation of virtual power plants and equitable profit distribution using shapely value theory. Mathematical Problems in Engineering 2018

[23] Wei, E. and Ozdaglar, A. (2013). On the o(1/k) convergence of asynchronous distributed alternating direction method of multipliers. In EEE Global Conference on Signal and Information Processing (IEEE)

[24] Weron, R. (2006). Modeling and Forecasting Electricity Loads and Prices (John Wiley & Sons Ltd). doi:10.1002/9781118673362

[25] Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. International Journal of Forecasting 30, 1030–1081. doi:10.1016/j.ijforecast.2014.08.008

[26] Neroni M. (2021). Ant Colony Optimisation with Warm-Up. Algorithms 14. Multidisciplinary Digital Publishing Institute.

[27] Omelčenko V., Manokhin V. (2021). Optimal Balancing of Wind Parks with Virtual Power Plants. Frontiers in Energy Research. doi: 10.3389/fenrg.2021.665295

[28] Podder, A. K., Islam, S., Kumar, N. M., Chand, A. A., Rao, P. N., Prasad, K. A., et al. (2020). Systematic Categorisation of Optimisation Strategies for Virtual Power Plants. Energies 13, 6251. doi:10.3390/en13236251

[29] Müller, F. L., Sundström, O., Szabó, J., and Lygeros, J. (2015). “Aggregation of Energetic Flexibility Using Zonotopes,” in 2015 54th IEEE Conference on Decision and Control (CDC), Osaka, Japan, 15-18 Dec. 2015 (IEEE), 6564–6569. doi:10.1109/CDC.2015.7403253

[30] Graabak, I., and Korpås, M. (2016). Balancing of Variable Wind and Solar Production in Continental Europe with Nordic Hydropower - A Review of Simulation Studies. Energ. Proced. 87, 91–99. doi:10.1016/j.egypro.2015.12.362.5th

[31] van Ackooij, W., Danti Lopez, I., Frangioni, A., Lacalandra, F., and Tahanan, M. (2018). Large-scale Unit Commitment under Uncertainty: an Updated Literature Survey. Ann. Oper. Res. 271, 11–85. doi:10.1007/s10479-018-3003-z

[32] Egieya, J., Čuček, L., Zirngast, K., Isafiade, A., and Kravanja, Z. (2020). Optimization of Biogas Supply Networks Considering Multiple Objectives and Auction Trading Prices of Electricity. BMC Chem. Eng. 2, 3. doi:10.1186/s42480-019-0025-5

[33] Ziegler, C., Richter, A., Hauer, I., and Wolter, M. (2018). “Technical Integration of Virtual Power Plants Enhanced by Energy Storages into German System Operation with Regard to Following the Schedule in Intra-day,” in 2018 53rd International Universities Power Engineering Conference (UPEC), Glasgow, UK, 4-7 Sept. 2018 (IEEE), 1–6. doi:10.1109/UPEC.2018.8541969

[34] Candra, D., Hartmann, K., and Nelles, M. (2018). Economic Optimal Implementation of Virtual Power Plants in the German Power Market. Energies 11, 2365. doi:10.3390/en11092365

[35] Duong, T. L., Duong, M. Q., Phan, V.-D., and Nguyen, T. T. (20202020). Optimal Reactive Power Flow for Large-Scale Power Systems Using an Effective Metaheuristic Algorithm. J. Electr. Comp. Eng. 2020, 1–11. doi:10.1155/2020/6382507

[36] Nguyen, T. T., Nguyen, T. T., Duong, M. Q., and Doan, A. T. (2020). Optimal Operation of Transmission Power Networks by Using Improved Stochastic Fractal Search Algorithm. Neural Comput. Applic 32, 9129–9164. doi:10.1007/s00521-019-04425-0

[37] Tan, Z., Li, H., Ju, L., and Tan, Q. (2018). Joint Scheduling Optimization of Virtual Power Plants and Equitable Profit Distribution Using Shapely Value Theory. Math. Probl. Eng. 2018, 3810492. doi:10.1155/2018/3810492

[38] Filippo, A. D., Lombardi, M., Milano, M., and Borghetti, A. (2017). “Robust Optimization for Virtual Power Plants,” in Advances in Artificial Intelligence. Lecture Notes in Computer Science (Cham: Springer), 10640, 41–50. doi:10.1007/978-3-319-70169-1_2

[39] Pandžić, H., Kuzle, I., and Capuder, T. (2013a). Virtual Power Plant Mid-term Dispatch Optimization. Appl. Energ. 101, 134–141. doi:10.1016/j.apenergy.2012.05.039

[40] Pandžić, H., Morales, J. M., Conejo, A. J., and Kuzle, I. (2013b). Offering Model for a Virtual Power Plant Based on Stochastic Programming. Appl. Energ. 105, 282–292. doi:10.1016/j.apenergy.2012.12.077

[41] Mashhour, E., and Moghaddas-Tafreshi, S. M. (2011). Bidding Strategy of Virtual Power Plant for Participating in Energy and Spinning Reserve Markets-Part I: Problem Formulation. IEEE Trans. Power Syst. 26, 949–956. doi:10.1109/tpwrs.2010.2070884

[42] Wegener, M., Villarroel Schneider, J., Malmquist, A., Isalgue, A., Martin, A., and Martin, V. (2021). Techno-economic Optimization Model for Polygeneration Hybrid Energy Storage Systems Using Biogas and Batteries. Energy 218, 119544. doi:10.1016/j.energy.2020.119544

[43] Vicentin, R., Fdz-Polanco, F., and Fdz-Polanco, M. (20192019). Energy Integration in Wastewater Treatment Plants by Anaerobic Digestion of Urban Waste: A Process Design and Simulation Study. Int. J. Chem. Eng. 2019, 1–11. doi:10.1155/2019/2621048

[44] Elkamel, M., Ahmadian, A., Diabat, A., and Zheng, Q. P. (2021). Stochastic Optimization for price-based Unit Commitment in Renewable Energy-Based Personal Rapid Transit Systems in Sustainable Smart Cities. Sust. Cities Soc. 65, 102618. doi:10.1016/j.scs.2020.102618

12. List of Abbreviations

ADMM	Alternating Direction Method of Multipliers
VPP	Virtual Power Plant
GI	Gradual Increase (a method of warm starts)
MPC	Model Predictive Control
RMPC	Robust Model Predictive Control
mSSA	Mulchannel Singular Spectrum Analysis
DER	Distributed Energy Resources